



UNIVERZITET U SARAJEVU  
ELEKTROTEHNIČKI FAKULTET  
ODSJEK ZA AUTOMATIKU I ELEKTRONIKU

---

# Razvoj sistema za vizualnu odometriju velike skalabilnosti

---

ZAVRŠNI RAD  
- DRUGI CIKLUS STUDIJA -

Autor:  
Emina Hasanović

Mentor:  
doc.dr Dinko Osmanković, dipl.ing.el.

Sarajevo,  
septembar 2020.

**Univerzitet u Sarajevu**

**Naziv fakulteta/akademije: Elektrotehnički fakultet**

**Naziv odsjeka: Odsjek za automatiku i elektroniku**

**Mentor: doc.dr Dinko Osmanković, dipl.ing.el.**

**Sarajevo, 16.09.2020.**

Postavka zadatka završnog rada II ciklusa:

## **Razvoj sistema za vizualnu odometriju velike skalabilnosti**

**Student:** Emina Hasanović

U radu je potrebno:

- objasniti pojam monokularne vizualne odometrije i načine rješenja datog zadatka
- objasniti korištene algoritme i alate pri realizaciji zadatka monokularne vizualne odometrije
- prikazati rezulata i realizaciju monokularne vizualne odometrije, te moguća unaprijeđenja ovog završnog rada

Preporučuje se realizacija algoritama u C++ programskom jeziku, uz korištenje Open Source Computer Vision (OpenCV) biblioteke.

### **Polazna literatura:**

[1] Scaramuzza D., Fraundorfer F., Siegwart R., "Real-Time Monocular Visual Odometry for On-Road Vehicles with 1-Point RANSAC", Computer Vision and Geometry Group, ETH, Zurich, 2009

[2] Tardif J.P., Pavlidis Y., Daniilidis K., "Monocular Visual Odometry in Urban Environments Using an Omnidirectional Camera", GRASP Laboratory, University of Pennsylvania, Philadelphia, 2008

---

doc. dr Dinko Osmanković, dipl. ing. el.

# **Potpisi članova Komisije za ocjenu i odbranu završnog rada II ciklusa studija**

---

prof. dr. Jasmin Velagić, dipl.el.ing,  
predsjednik komisije

---

doc. dr. Dinko Osmanković, dipl.el.ing,  
mentor

---

prof. dr. Bakir Lačević, dipl.el.ing,  
član komisije

---

# Zahvalnica

*Veliku zahvalnost dugujem svom mentoru doc.dr Dinku Osmanković, na izuzetnoj pomoći prilikom izrade rada, na posvećenom vremenu, nesebičnom zalaganju i savjetima koji su mi olakšali izradu ovog rada.*

*Veliku zahvalnost dugujem svojoj porodici, roditeljima i bratu za ohrabrenje i neizmjernu podršku tokom cijelog obrazovanja.*

*Hvala i svim prijateljima na podršci i razumijevanju.*

Emina Hasanović,  
septembar 2020.

---

# Izjava o autentičnosti rada

Seminarski rad, završni (diplomski odnosno magistarski) rad za I i II ciklus studija i integrirani studijski program I i II ciklusa studija, magistarski znanstveni rad i doktorska disertacija<sup>1</sup>.

Ime i prezime: Emina Hasanović

Naslov rada: Razvoj sistema za vizualnu odometriju velike skalabilnosti

Vrsta rada: Završni rad za II ciklus studija

Broj stranica: 50

Potvrđujem:

- Da sam pročitao/la dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- Da sam svjestan/na univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- Da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- Da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- Da sam jasno naznačio/la prisustvo citiranog ili parafraziranog materijala i da sam se referirao/la na sve izvore;
- Da sam dosljedno naveo/la korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- Da sam odgovarajuće naznačio/la svaku pomoć koju sam dobio/la pored pomoći mentora/ice i akademskih tutora/ica.

Mjesto, datum: \_\_\_\_\_

Potpis kandidata: \_\_\_\_\_

Potpis mentora: \_\_\_\_\_

---

<sup>1</sup>U radu su korišteni slijedeći dokumenti: Izjava autora koju koristi Elektrotehnički fakultet u Sarajevu; Izjava o autentičnosti završnog rada Centra za interdisciplinarnе studije – master studij „Evropske studije”, Izjava o plagijarizmu koju koristi Fakultet političkih nauka u Sarajevu.

## Sažetak

Ova master teza obrađuje razvoj i implementaciju algoritma monokularne vizualne odometrije za lokalizaciju robota ili vozila. Značajnost mobilnih roboata je izrazito porasla nekoliko posljednjih godina. Za ove roboate suštinski je bitno poznavati njihovu lokaciju u okolini. Vizualna odometrija je metod za estimiranje kretanja jedne ili više kamera, korištenjem samo vizualnih podataka ovih kamera. U ovom radu korištena je jedna kamera. Linijske značajke se detektuju sa tri različita detektora: Line Segment Detector, Standard Hough Line Transform and Probabilistic Hough Line Transform. Algoritam je implementiran u C++ programskom jeziku koristeći biblioteku kompjuterske vizije OpenCV. Rezultati algoritma su provjereni na KITTI skupu podataka, jer KITTI podaci sadrže informaciju o stvarnoj trajektoriji vozila/roboata. Na kraju, nekoliko je eksperimenata sprovedeno da bi se odredila tačnost algoritma vizualne odometrije koji je razvijen u ovom radu.

**Ključne riječi:** Monokularna vizualna odometrija, Lokalizacija, Kamera i slike

## Abstract

This thesis is about the development and implementation of a monocular visual odometry algorithm for robot or vehical localisation. The relevance of mobile robots has distinctly increased during the past years. For these robots it is essential to know their exact location in the environment. Visual odometry is the method of calculating the egomotion of one or multiple cameras, by only using the visual data provided by these cameras. In this approach, a monocular camera is used. The line features get detected with three different line detectors: Line Segment Detector, Standard Hough Line Transform and Probabilistic Hough Line Transform. The algorithm will be programmed in C++ using the computer vision library OpenCV. The results are reported on the KITTI dataset since KITTI dataset provides information about ground truth trajectory. In the end, various experiments are performed to determine the accuracy of the visual odometry algorithm developed in this work.

**Keywords:** Monocular Visual Odometry, Localization, Camera and Frames

# Sadržaj

<b>Popis slika</b>	<b>ix</b>
<b>Popis tabela</b>	<b>x</b>
<b>Indeks pojmova</b>	<b>x</b>
<b>1. Uvod</b>	<b>1</b>
1.1. Obrazloženje teme . . . . .	1
1.2. Pregled stanja u oblasti istraživanja . . . . .	2
1.3. Motivacija . . . . .	3
1.4. Osnovni ciljevi i plan rada . . . . .	4
1.5. Struktura rada . . . . .	5
1.6. Očekivani rezultati . . . . .	5
1.7. Zaključak . . . . .	5
<b>2. Osnovni koncepti kompjuterske vizije</b>	<b>6</b>
2.1. Formiranje slike i model kamere . . . . .	6
2.1.1. Pinhole model kamere . . . . .	6
2.1.2. Transformacije koordinata . . . . .	7
2.2. Kalibracija kamere . . . . .	9
2.2.1. Intrinsični i ekstrinsični parametri kamere . . . . .	9
2.3. Geometrija dvaju pogleda . . . . .	11
2.3.1. Epipolarna geometrija . . . . .	11
2.4. Zaključak . . . . .	15
<b>3. Monokularna vizualna odometrija</b>	<b>16</b>
3.1. Prikupljanje slijeda ulaznih slika iz videa . . . . .	16
3.2. Vađenje (ekstrakcija) značajki sa slike . . . . .	17
3.2.1. Line Segment Detector . . . . .	18
3.2.2. Hough Line Transform . . . . .	20
3.3. Korespondencija značajki . . . . .	22
3.3.1. Kanade-Lucas-Tomasi (KLT) metod praćenja značajki . . . . .	24
3.4. Estimacija pozicije kamere . . . . .	25
3.4.1. Odstranjivanje vanpopulacijskih značajki . . . . .	26
3.4.2. Procjena matrice rotacije i translacije . . . . .	28
3.5. Zaključak . . . . .	29

<b>4. Programska implementacija</b>	<b>30</b>
4.1. Softversko rješenje . . . . .	30
4.1.1. OpenCV . . . . .	31
4.1.2. KITTI podaci . . . . .	33
4.2. Implementacija algoritma . . . . .	35
4.3. Eksperimentalni rezultati . . . . .	37
4.4. Zaključak . . . . .	43
<b>Literatura</b>	<b>47</b>

# Popis slika

2.1	Pinhole model kamere [1] . . . . .	6
2.2	Ilustracija transformacije koordinata između sistema kamere i okoline [2] . .	8
2.3	Ispupčena (lijevo) i udubljena(desno) distorzija [3] . . . . .	10
2.4	Geometrija dvaju pogleda [2] . . . . .	12
3.1	Aperture problem [4] . . . . .	17
3.2	Gradijent i linije promjene [5] . . . . .	18
3.3	Linijski dijelovi [5] . . . . .	19
3.4	Pravougaona aproksimacija linijskog regiona i poravnate tačke [5] . . . . .	19
3.5	Primjer detekcije linija pomoću LSD detektora [6] . . . . .	19
3.6	Preslikavanje linije u Houghov prostor [7] . . . . .	20
3.7	Transformacija dvije tačke $p_o$ i $p_1$ (lijeva slika) u dvije linije u Hough-vom prostoru (desna slika). Presijecanje dvije linije u Houghvom prostoru ukazuje na liniju koja prolazi kroz obje tačke $p_o$ i $p_1$ [7] . . . . .	21
3.8	Primjer detekcije linija SHLT (lijeva slika) i PHLT (desna slika) [8] . . . . .	22
3.9	Primjer praćenja značajki pomoću Kanade-Lucas-Tomasi metode [9] . . . . .	23
3.10	Primjer uparivanja značajki na dvije slike pomoću Flann-Matcher tehnike [10] . . . . .	24
3.11	Primjer RANSAC metode [11] . . . . .	27
4.1	Struktura OpenCV-a . . . . .	32
4.2	Platforma za snimanje podataka: VW Passat sa senzorima [12] . . . . .	33
4.3	Raspored senzora na autu VW Passat: Dimnizije i mesta gdje su senzori postavljeni u odnosu na ram vozila (crveno), te udaljenost od zemlje je označena zelenom bojom. Transformacije između senzora su prikazane plavom bojom [12] . . . . .	34
4.4	Značajke i njihovo pomjeranje: LSD detektor . . . . .	35
4.5	Značajke i njihovo pomjeranje: PHLT detektor . . . . .	36
4.6	Značajke i njihovo pomjeranje: SHLT detektor . . . . .	36
4.7	Iscrtavanje trajektorije kretanja kamere na prozoru u x-z ravni. S lijeva na desno: LSD detektor, PHLT detektor i SHLT detektor . . . . .	36
4.8	Trajektorije u x-z ravni za različit broj frejmova 200, 500, 1000 i 2000 respektivno . . . . .	38
4.9	Ukupna trajektorija u x-z ravni . . . . .	39
4.10	Relativna rotacijska greška izražena u $[^\circ/m]$ u odnosu na različit broj frejmova	40
4.11	Relativna translacijska greška izražena u $[\%]$ u odnosu na različit broj frejmova . . . . .	41
4.12	Translacijska greška izražena u $[m]$ po pojedinim osama za sve detektore . .	42
4.13	Rotacijska greška izražena u $[^\circ]$ po pojedinim uglovima za sve detektore . .	42

# Popis tablica

4.1	ATE i relativne greške za 200 frejmova . . . . .	39
4.2	Vremena izvršavanja algoritama sa LSD, PHLT i SHLT detektorima . . . .	41

# Poglavlje 1.

## Uvod

U 20. stoljeću veliki je interes i potreba za automatizacijom mnogih poslova. Jedan od pristupa ovom problemu su autonomni inteligentni sistemi koji mogu raditi duže vrijeme bez ljudske kontrole. Mnogi su primjeri ovakvih sistema, istraživanje svemira gdje je ljudsko upravljanje znatno otežano, autonomna kolica za nepokretne, kućanska poma-gala, mnoge su primjene i u automobilskoj industriji. Ovi sistemi na neki način moraju razumjeti okolinu u kojoj se nalaze i kreću, da bi mogli raditi bez ljudske kontrole. Odometrija objašnjava kako iskoristiti ulazne podatke za procjenu položaja i kretanja objekta. Jedan od načina da se to ostvari, a ujedno je invarijantan na vrstu pogona, je **vizualna odometrija** (eng. *Visual Odometry - VO*) [13].

### 1.1. Obrazloženje teme

Odometrija je postupak korištenja podataka o kretanju za procjenu relativnog položaja objekta. Kod vizualne odometrije podaci korišteni za takvu procjenu su isključivo slike ili video sekvenca. Takva odometrija nije ovisna o načinu kretanja objekta, već se može upotrijebiti neovisno o pogonu vozila ili robota. Dakle, vizualna odometrija vrši procjenu pozicije i usmjerenja kamere iz slijeda slike s njenog izlaza. Metode korištene u ovom radu su većinom prilagođene za vizualnu odometriju za cestovna vozila, i to najviše vozila koja se kreću u gradskim ulicama [14]. Naime, u takvom okruženju vozilo se kreće ulicom uz niz zgrada koje ga okružuju sa obje strane, nailazi na pješake, te susreće se sa drugim vozilima i objektima. Cilj ove primjene vizualne odometrije je procijeniti kretanje vozila na cesti, te predvidjeti prostor i kretanje svih drugih objekata koji se nalaze uz cestu ili na cesti. Ovo je primjer monekularne odometrije, što implicira da je u postupku procjene kretanja korištena samo jedna kamera koja je bila montirana na vozilu. Naravno mogu biti korištene dvije kamere, to je onda stereo vizualna odometrija ili više njih. Oba načina imaju i prednosti i nedostatke, te će to biti pojašnjeno u nastavku ovog rada.

Iz navedenih primjera je moguće zaključiti zašto je ova tema pogodna za istraživanje i koliko su istraživanja ove teme i sličnih oblasti jako korisna ljudima. U nekoliko posljednjih decenija istraživanja su usmjerena upravo prema ovoj oblasti i time se bave naučnici iz cijelog svijeta, te su zabilježeni veliki uspjesi, napredak, ali i različiti pristupi kako se ovaj zadatak može obaviti. Danas roboti obavljaju komplikovane zadatke, dok su prije to većinom radili ljudi i ljudska interakcija je bila neophodna. Ti poslovi su često bili opasni za čovjeka, pa i nedostupni čovjeku, poslovi i zadaci u oblasti hemijske industrije, vojske,

svemirske ekspedicije i istraživanja. Prve ideje su se javile upravo u oblasti svemirskih istraživanja i mogućnost operiranja mobilnih robova na Marsu, tj. procjene položaja Mars Rover-a. Ideja za korištenje vizualne odometrije je upravo proizašla iz činjenice da na Marsu nema globalnog sistema za pozicioniranje (eng. *Global Positioning System - GPS*), a i točkovna odometrija je nepogodna zbog pješčanog terena uslijed proklizivanja koje unosi nesistemsku grešku [13]. Jedna od prednosti vizualne odometrije u odnosu na točkovnu i inercijalnu odometriju jeste upravo to što se može koristiti za bilo koje vozilo (zračno, podzemno, podvodno) i potrebni su relativno jeftini senzori/kamere u odnosu na IMU (eng. *Inertial Measurement Unit - IMU*). Svakako jedna od manih vizualne odometrije je da računarski skupa i zahtijevna, te se može koristiti samo na područjima sa dobrim osvjetljenjem i sa dobrom teksturom. Sva ova prva istraživanja i primjene u svemiru su podstakla da se autonomna vozila i roboti primjene i u drugim oblastima ljudskog života, te danas imamo njihovu zastupljenost u skoro svim granama industrije, svakodnevničici, pa i za zabavu [13].

## 1.2. Pregled stanja u oblasti istraživanja

U literaturi već postoje različiti pristupi lokalizacije vozila i mobilnih robova. Kao što je već ranije navedeno, GPS je najpopularniji način određivanja vlastite pozicije. Uglavnom se koristi za navigaciju u otvorenim prostorima i radi na principu primanja informacije sa satelita [15]. Naime, ukoliko je signal izgubljen, GPS neće uspjeti, što se može lako desiti ako vozilo uđe u neku zatvorenu mjesto kao što je zgrada ili tunel. Mobilni robot može biti namijenjen i za operiranje u unutrašnjim prostorima, pa GPS nije moguće koristiti za obavljanje zadatka lokalizacije u ovom slučaju.

Za unutrašnju navigaciju koriste se laserski skeneri, koji se često koriste za kreiranje mape okruženja sa visokom rezolucijom tako što skeniraju prostor unutar svog radnog opsega. Ovi LIDAR (eng. "*light*" and "*radar*") sistemi šalju laserske zrake u svim smjerovima da bi izmjerili svoju distancu od objekata koji ih okružuju tako što analiziraju reflektirane zrake [16]. Ove informacije se dalje koriste za određivanje vlastite pozicije u prostoriji.

Postoji još jedan popularan pristup estimacije vlastitog kretanja koristeći IMU (eng. *Inertial Measurement Unit*) koji se inače sastoje od akcelerometra i žiroskopa. Integriranjem ubrzanja u vremenu, oddređuje se trenutna brzina i pozicija. Pozicija i orientacija su izračunate dodavanjem trenutnih izmjerjenih vrijednosti prethodno estimiranoj poziciji [17]. Upravo zato se greška pri mjerenu povećava vremenom korištenjem ove tehnike.

Nadalje, postoji još jedan pristup određivanja vlastite pozicije koji se temelji na prethodnom i trenutnom kretanju. Vizualna odometrija koristi kamеру kao senzor za mjerjenje, te se primjenjuju različiti algoritmi za procesiranje slike za samolokalizaciju. Procesiranje slike i kompjuterska vizija su relativno nova područja istraživanja, ali već je objavljeno mnogo korisnih koncepata i implementacija. Jedna od istaknutih publikacija je svakako i spomenuti Mars Rover od strane NASA-e (eng. *National Aeronautics and Space Administration - NASA*) [18]. Među prvim istraživanjima su i istraživanja Hansa Moraveca 1980-tih godina gdje je on koristio jednu kliznu kamерu da bi estimirao kretanje mobilnog robova/rovera u zatvorenim područjima [19]. To je bio stereo pristup (klizna

kamera bi uslikala jednu sliku u svojoj orginalnoj poziciji i drugu sliku pomjeranjem u drugu poziciju). Odvojeni detektor značajki koji je nazvan Moravec detektor ivica je razvijen kako bi pratio tačke kroz nekoliko slika/frejmova.

Problem određivanja relativne pozicije kamere i 3D strukture od seta seta slika je mnogo izučavano tokom proteklih godina od strane društva za kompjutersku viziju poznato kao struktura iz pokreta (eng. *Structure for Motion* - SfM). Uspješni rezultati su prikupljeni koristeći samo jednu kameru na velikim distancama (nekoliko stotina metara pa i kilometara) u posljednjoj deceniji koristeći i perspektivnu i omnidirekionalnu kameru. U nastavku će biti navedeni neki od tih radova [20].

Prethodni radovi mogu biti podijeljeni u tri kategorije: metod baziran na značajkama, metod baziran na izgledu i hibridni metod. Metod baziran na značajkama je zapravo baziran na istaknutim i ponavljujućim značajkama koji su praćeni kroz frejmove/slike. Metod baziran na izgledu koristi samo informaciju o intenzitetu cijele slike ili njenih subregija. Hibridni metod je kombinacija prethodna dva. Radovi koji su bazirani na prvoj kategoriji su radovi [21], [22], [23] i [24]. U [21], Bosse i ostali istraživači su koristili iščezavajuće tačke i 3D linije da bi odredio strukturu i kretanje na 946 metara dugoj putanji. U [24] Nister i ostali autori su koristili stereo pristup ali su dali i monokularno rješenje implementirajući cijelu strukturu iz algoritma za kretanje koji koristi prednost algoritma sa pet tačaka i RANSAC robusnu estimaciju. U [22] Corke i ostali su predstavili dva pristupa monokularnoj vizualnoj odometriji baziranoj na omnidirekisionalnim slikama sa katadioptrične kamere. Njihov eksperiment je bio namijenjen za planetarnog rovera pa su izvodili eksperimente u pustinji i koristili su značajke sa zemljine ravnine. U [23] Lhuillier je koristio RANSAC sa pet tačaka i tzv. bundle adjustment da bi odredio oboje, kretanje i 3D mapu. Što se tiče hibridnog metoda, on se spominje u radovima od [25], [26] i [27]. U [25] Goecke i ostali autori su koristili Fourier-Mellin transformaciju za registriranje perspektivnih slika ravnine tla slikanih iz auta. Rezultati su prikazani na putanji od 300 metara. U [26] Milford i ostali istraživači predstavili su metod za ekstrakciju približne rotacione i translacione brzine sa jedne prespektivne kamere montirane na automobilu što je kasnije korišteno za generiranje koherentne mape urbanog okruženja. Međutim metod baziran na izgledu i takvi pristupi problemu nisu robusni na okluzije. Zbog ovog razloga u radovima [28] i [29] korišten je izgled okoline da bi se estimirala rotacija automobila i značajke sa podloge za estimaciju translacije i faktora za skaliranje.

Usko povezana tema sa vizualnom odometrijom je i uzastopno lokaliziranje i mapiiranje (eng. *Simultaneous Localization and Mapping* - SLAM) čiji je cilj pored estimiranja kretanja robota i uporedno građenje i ažuriranje mape okruženja. SLAM se često izvodi koristeći i druge senzore pored regularne kamere, mada je su posljednjih godina postignuti izvrsni rezultati koristeći samo kamere.

### 1.3. Motivacija

U današnje vrijeme nauka je posvećena izučavanju autonomnih vozila ili robota, te postaju sve zastupljeniji u svakodnevnom životu ljudi. Ovaka vozila imaju jako velike mogućnosti i imaju mnogobrojne primjene. Jedno polje primjene je istraživanje područja

koja su teško dostupna čovjeku ili opasna po čovjeka. Naprimjer, autonomno vozilo za kretanje po području koje je zahvatio požar ili na kojem se desila nuklearna katastrofa, vozilo namijenjeno za istraživanje drugih planeta, pomoći pri kretanju starijih ljudi i invalida itd. U novije vrijeme automobilska industrija nastoji lansirati na tržište i samoupravljeni automobil što bi znatno smanjilo i broj automobilskih nesreća. Iz prethodno navedenih razloga vidimo da je potrebno napraviti autonomno vozilo sa jako dobrom performansom.

Najznačajniji aspekt pri ovom razvoju je robusno i tačno samolokaliziranje ovih roboti. Tačna lokalizacija robota je potrebna za navigaciju. Postoje mnogi načini esitmacije kretanja ovih roboti. Najčešća tehnika je GPS (eng. *Global Positioning System*). GPS jako dobro radi pri kretanju vozila u vanjskom prostoru dok za unutrašnji prostor ne može biti korišten. Također, tačan je samo unutar nekoliko metara, pa nije baš prikladan za svaku aplikaciju. Druge dvije opcije za lokalizaciju su *senzori za mjerjenje brzine točkova* robota, te *IMU* (eng. *Inertial Measurement Unit*). Međutim, pouzdan IMU je dosta skup, dok korištenje senzora sa točkova jeste pouzdano ukoliko nema proklizavanja, što često nije slučaj.

Ljudi kao senzor za orijentaciju koriste svoje oči, pa se slično može i kamera iskoristiti za lokalizaciju robota. Proces određivanja samokretanja robota iz sekvence slika je, kao što je rečeno, *vizualna odometrija*. Kamere postaju sve tačnije i jeftinije u proteklih nekoliko godina, pa su postale široko dostupan senzor na robotima. Ovaj završni rad je upravo je implementacija jednog pristupa vizualne odometrije za samolokalizaciju.

## 1.4. Osnovni ciljevi i plan rada

Kao što je ranije navedeno cilj ovog rada je razvoj i implementacija algoritma monokularne vizualne odometrije za samolokalizaciju vozila/mobilnog robota. Nakon što je objašnjena osnova problema dizajnirani VO algoritam će biti ilustrovan i diskutovan detaljno. U narednim poglavljima će biti prikazano na kojem skupu podataka će algoritam biti testiran i detaljno objašnjene samog skupa podataka. Naravno, bit će predviđeni eksperimentalni rezultati i koja je tačnost dizajniranog algoritma. Detaljno će biti dat uvid u sve korištene algoritme, pogotovo algoritme za ekstrakciju i praćenje značajki sa slikama, algoritme za određivanje matrica rotacije i translacije, te faktora za skaliranje, kako će faktor biti izračunat zbog činjenice da je dubina slike nepoznata jer je korištena jedna kamera. Također, navest će se svi korišteni alati i paketi, programski jezi, na koji način su korišteni i zašto su oni pogodni za realizaciju ovog završnog rada. Spomenute značajke na slikama i algoritmi vezani za njihovu ekstrakciju su prvi i esencijalni korak ka rješenju. Ovaj problem i zadatak je jedan od najpopularnijih posljednjih decenija, tj. sama robotska vizija. Kako je čovjeku njegov vid i oči kao senzor najbitniji za kretanje, tako je usavršavanjem kamere i povećavanjem rezolucije sada moguće sličnu ideju implementirati i na robotskom sistemu, potrebno je da se ovi algoritmi sve više razvijaju, postaju brži, efikasniji, jeftiniji i sl. Očekujemo da će ovaj rad doprinijeti povećanju znanja u oblasti istraživanja i svakako čitaocu rada dati detaljan uvid u ovaj dio robotske vizije.

## 1.5. Struktura rada

U ovom poglavlju bit će dat kratak pregled metoda koje će se koristiti u radu i pregled svakog koraka koji je važan za dizajn algoritma monokularne vizualne odometrije. Završni rad će biti implementiran u OpenCV (biblioteka za kompjutersku viziju), u C++ programskom jeziku. Ovaj završni rad će dati sve informacije o tome kako je pozicija i orijentacija kamere određena. Rad se sastoji od sljedećih poglavlja:

1. Prvo poglavlje daje kratki uvod o temi, dosadašnja istraživanja i rezultate, te ilustrira strukturu teze.
2. Drugo poglavlje opisuje osnovne koncepte koji se koriste u kompjuterskoj viziji, informacije o slikama i formirajući slike, kameri i parametrima kamere.
3. Glavno poglavlje, treće poglavlje će dati detaljno objašnjenje cijelog algoritma i dizajn rješenja.
4. U četvrtom poglavlju bit će prikazana programska implementacija i eksperimentalni rezultati, te kakve su performanse algoritma.
5. Zadnje poglavlje je zaključak cijelog rada i predlaže dalja moguća unaprijeđenja rada.

## 1.6. Očekivani rezultati

Na kraju rada odredit će se tačnost, preciznost i robusnost implementiranog algoritma monokularne vizualne odometrije i bit će priloženi eksperimentalni rezultati. Očekuje se da se implementirani VO algoritam može koristiti za samolokalizaciju robota ili vozila i da je tačno poznato s kojom greškom je estimirana trajektorija kretanja vozila ili robota. Očekuje se da je greška algoritma u odnosu na stvarnu trajektoriju kretanja robota ili vozila vrlo mala za translaciju i rotaciju, te da je tačno poznato koji aspekt algoritma unosi grešku.

## 1.7. Zaključak

U uvodnom poglavlju je opisana i definirana tema ovog završnog rada, dosadašnja istraživanja, motivacija autora, osnovni ciljevi i plan rada, te metodologija rada i očekivani rezultati. U narednom poglavlju će biti opisani osnovni koncepti kompjuterske vizije. Naime, bit će objašnjeno kako izgleda proces formiranja slike, koji je to osnovni model kamere, kako se vrši kalibracija kamere i zašto su parametri kamere bitni koji se dobiju pri kalibraciji. Na kraju je opisano kako se je moguće dobiti koordinate stvarne tačke iz okoline koristeći projekciju te tačke na ravninu slike ili više slike, te kako je moguće pratiti kretanje kamere na osnovu niza slika. Date su su osnovne matematičke relacije.

## Poglavlje 2.

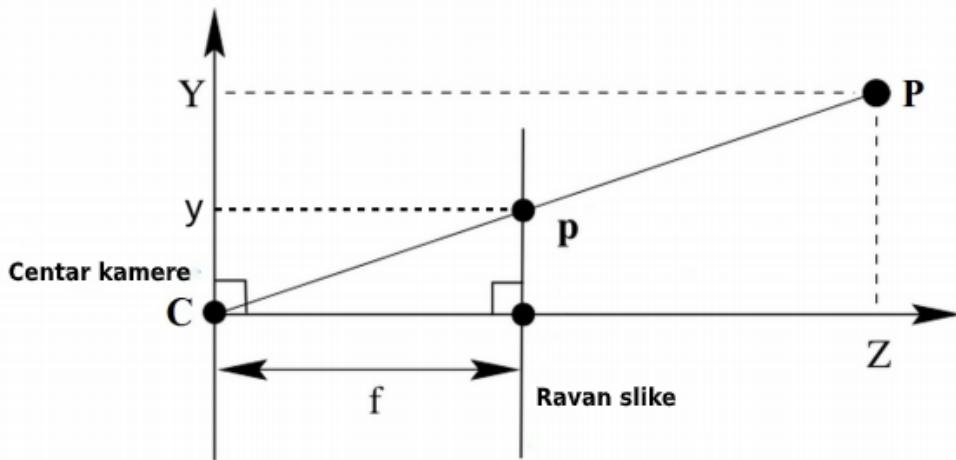
# Osnovni koncepti kompjuterske vizije

U ovom poglavlju će biti predstavljene osnovne tehnike za obradu slike, općenito. Počinje sa predstavljanjem modela kamere i kako se slika formira. Neki osnovni matematički koncepti će također biti predstavljeni.

### 2.1. Formiranje slike i model kamere

#### 2.1.1. Pinhole model kamere

Temelj svih koncepata kompjuterske vizije je kamera i slike sa kamere. Ukoliko želimo da koristimo ove slike za neku aplikaciju potrebno je znati šta su one ustvari. Na početku objašnjen je proces formiranja stvarnosti u slici. Objasnjen je tzv. pinhole model kamere [1]. Slika 2.1 objasnjava kakav je to model.



Slika 2.1: Pinhole model kamere [1]

Senzor kamere detektuje zrake koje se odbijaju od 3D tačke  $P$ . Slika tačke je formirana na ravnini kamere. Zbog jednostavnosti prvo će biti objašnjen 2D primjer. Tačka  $P = (Y, Z)^T$  sa udaljenosću  $Z$  od centra kamere i pozicija  $Y$  je mapirana u tačku  $p = (y)$  na slici. Fokus

kamere je tačka  $f$ , pa imamo da je (kao što je poznato iz optike) [1]:

$$\frac{y}{Y} = \frac{f}{Z} \quad (2.1)$$

U ovoj formuli se pretpostavi da je fokus kamere vrlo mali u odnosu na udaljenost  $Z$  tačke  $P$  od kamere. Ova pretpostavka vrijedi za mnoge realne situacije, s obzirom da je fokus uvećim samo nekoliko centimetara. Pa se transformacija stvarnosti (3D) u sliku (2D) može napisati kao:

$$(X, Y, Z) \rightarrow \left( \frac{f}{Z} X, \frac{f}{Z} Y \right) \quad (2.2)$$

Ovaj model se naziva idealnih pinhole modelom. U nastavku rada bit će korištene homogene koordinate zbog jednostavnosti, koje se mnogo koriste u projektivnoj geometriji. Ovo znači da je standardnim koordinatama koja sadrži faktor skaliranja za ovu tačku. S obzirom na to 2D koordinata u Kartezijevom koordinatnom sistemu  $(x, y)$  će biti  $(u, v, w) = (wx, wy, w)$  u homogenoj reprezentaciji. Nadalje, jednačina 2.1 povezuje tačku u stvarnosti sa tačkom na slici kamere, koristeći koordinatni sistem kamere. Za obradu slike bitno je iskoristiti koordinate slike u pikselima koji počinje sa  $(0,0)$  u lijevom desnom uglu slike. Da bismo preveli centar optičke ose u centar koordinata slike, imat ćemo offset  $(o_x, o_y)$  za x i y koordinate. Na kraju jednačina koja transformira stvarnu 3D tačku  $X$  u 2D tačku  $x$  je:

$$\lambda \cdot \mathbf{x} = \mathbf{K} \cdot \mathbf{I} \cdot \mathbf{Y} \quad (2.3)$$

gdje je  $\mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ ,  $\mathbf{K} = \begin{pmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{pmatrix}$ ,  $\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ ,  $\mathbf{Y} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$ .

$\mathbf{K}$  je matrica parametara kamere, a  $\mathbf{I}$  jedinična matrica dimenzija  $3 \times 4$ , koja je bitna kod određivanja krutog kretanja kamere.  $\lambda$  je skalirajući faktor tačke slike. Za sada, imamo jednačinu koja mapira stvarnu tačku u 2D tačku slike kamere [1].

### 2.1.2. Transformacije koordinata

U mnogo slučajeva potrebno je definirati stvarne koordinate relativno u odnosu na stvarno okruženje, a ne u odnosu na frejm kamere. Naprimjer, pozicija robota treba da je estimirana relativno u odnosu na sobu u kojoj se robot kreće, a ne relativno u odnosu na kameru koja snima. U tom slučaju, potrebno je da se koordinatni sistem kamere transformira u koordinatni sistem okruženja. To se naziva euklidska transformacija. Kada se rade takve transformacije, postoji šest stepeni slobode: prvo, sistem se može pomjerati, translirati,

po x, y i z osi. Ove translacije su opisane  $3 \times 1$  vektorom translacije  $\mathbf{t}$ . Nadalje, koordinatni sistem se može rotirati oko sve tri ose [30]. Takva rotacija se opisuje  $3 \times 3$  matricom  $\mathbf{R}$ .

$$\mathbf{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \quad \mathbf{R} = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \quad (2.4)$$

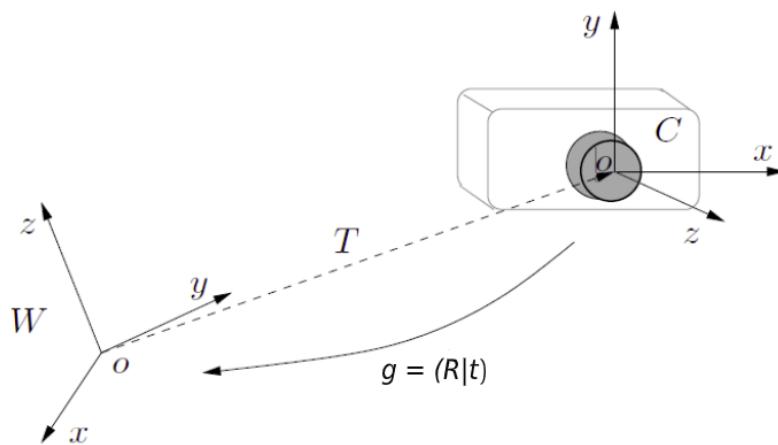
Matrica rotacije  $\mathbf{R}$  je dobijena tako što su napravljene tri uzastopne rotacije oko koordinatnih osa za uglove  $\Phi$ ,  $\Theta$  i  $\Psi$ . Ova tri rotaciona ugla  $\Phi$ ,  $\Theta$  i  $\Psi$  formiraju rotacijski vektor  $\mathbf{r}$ . Međutim, za većinu proračuna koristi se  $3 \times 3$  matrica rotacije  $\mathbf{R}$ . Uz pomoć Rodrigues-ove formule, moguće je transformisati vektor  $\mathbf{r}$  u matricu  $\mathbf{R}$  i obrnuto. Članovi  $r_1, \dots, r_9$  matrice rotacije su izračunati množenjem svake matrice rotacije oko pojedinačnih osa:

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \Phi & \sin \Phi \\ 0 & -\sin \Phi & \cos \Phi \end{pmatrix} \cdot \begin{pmatrix} \cos \Theta & 0 & -\sin \Theta \\ 0 & 1 & 0 \\ \sin \Theta & 0 & \cos \Theta \end{pmatrix} \cdot \begin{pmatrix} \cos \Psi & \sin \Psi & 0 \\ -\sin \Psi & \cos \Psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.5)$$

Kada imamo matricu rotacije i vektor translacije tačka  $X_c$  u koordinatnom sistemu kamere može biti transformirana u tačku  $X_w$  u odnosu na koordinatni sistem okoline, ali primijenimo:

$$X_w = \mathbf{R}X_c + \mathbf{t} \quad (2.6)$$

Slika 2.2 pokazuje upravo te transformacije.



**Slika 2.2:** Ilustracija transformacije koordinata između sistema kamere i okoline [2]

U homogenoj reprezentaciji, transformacija  $g = (R|t)$ , može biti napisano na sljedeći

način:

$$\mathbf{g} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.7)$$

Ukoliko poznajemo sve ove transformacije one mogu biti primjenjene na jednačinu projekcije kamere 2.3 [2]. Pa kada mijenjamo  $\mathbf{I}$  sa  $\mathbf{g}$  konačno dobijemo vezu između 3D koordinata tačke  $X_o = (X_o, Y_o, Z_o)$ , koja je u odnosu na koordinatni sistem okoline, a ne koordinatni sistem kamere, i tačke koja je projicirana na sliku  $x_o = (x_o, y_o)$ :

$$\lambda \cdot \begin{pmatrix} x_o \\ y_o \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \end{pmatrix} \cdot \begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix} \Leftrightarrow \lambda \cdot x_o = \mathbf{K} \cdot (\mathbf{R}|t) \cdot X_o \quad (2.8)$$

## 2.2. Kalibracija kamere

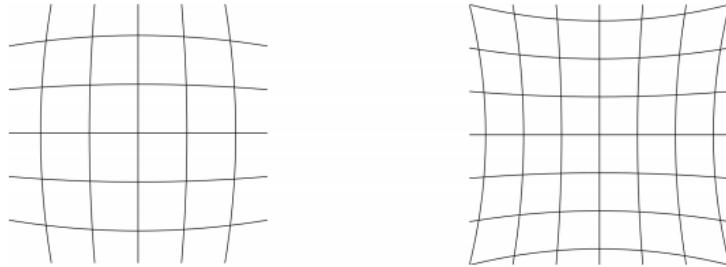
### 2.2.1. Intrinsični i ekstrinsični parametri kamere

U prethodnom poglavlju pokazano je kako se neka tačka iz okoline formira na slici. Da bi se uradio ovaj proračun, matrica kamere  $\mathbf{K}$ , koja se sastoји od vrijednosti za fokalnu dužinu ( $f$ ) i optičkog centra ( $o_x, o_y$ ), mora biti poznata. Nadalje, transformacije koordinata ( $\mathbf{R}|t$ ) se moraju odrediti, da bi se moglo odrediti kako je sistem kamere postavljen u odnosu na sistem okoline. U ovom kontekstu, parametri kamere su zapravo intrinsični parametri kamere, jer opisuju određene vrijednosti unutar određene kamere. ( $\mathbf{R}|t$ ) vrijednosti su ekstrinsični parametri kamere, s obzirom da oni određuju eksternu pozu kamere u odnosu na okolinu.

Intrinsični i ekstrinsični parametri kamere se određuju kalibracijom kamere. Da bi se kalibracija kamere uradila potrebno je uslikati poznati objekat koji se naziva kalibracijski prsten. U većini slučajeva se koristi šahovska ploča. Algoritam za detekciju uglova pronađe sve uglove bijelih i crnih kvadrata šahovske ploče na slici. Ovo generira set tačaka slike  $x_i$  koji su korespondenti sa vrhovima uglovima šahovske ploče. Nadalje, veličina šahovske ploče i njenih kvadrata mora biti poznata, tako da se može odrediti set stvarnih koordinata vrhova uglova šahovske ploče  $X_i$ . Obično se u donji lijevi ugao postavi centar koordinatnog sistema "okoline". U ovom slučaju svi vrhovi će imati koordinate  $(X_i, Y_i, 0)$ . Kada su poznata dva seta tačaka slike  $x_i$  i odgovarajućih stvarnih tačaka  $X_i$  mogu biti izračunati i intrinsični i ekstrinsični parametri. Ovo se uradi kada se sve vrijednosti uvrste u jednačinu 2.8 i jednačina se riješi se po svim nepoznatim parametrima [31].

U prethodnom proračunu koristi se idealni pinhole model kamere. U stvarnosti ne postoji rupa u projekcijskom centru kamere, nego tanka leća. Tanku leću dovodi do distorzije na slici. Postoje različiti tipovi distorzije koji mogu modificirati sliku. Najbitniji efekti su

udubljena i ispučena distorzija [3]. Na slici 2.3 se vidi kako one izgledaju.



Slika 2.3: Ispučena (lijevo) i udubljena (desno) distorzija [3]

Efekat distorzije slike na njenim krajevima je naročito izražen, utiče na njeno formiranje i taj efekat se mora uzeti u obzir pri proračunima koji povezuju tačke na slici i stvarne tačke. Uvodi se tzv. koeficijent distorzije  $k$  u jednačine koje povezuju tačke slike i stvarne tačke. Ovaj koeficijent spada u intrinsične parametre kamere i bude proračunat tokom kalibracije. Nelinearna optimizacija sa Levenberg-Marquardt algoritmom pronalazi matričnu kamere  $\mathbf{K}$ , koeficijente distorzije  $k$ , transformacije koordinata  $(\mathbf{R}|\mathbf{t})$ , te minimizira grešku. Dakle, algoritam estimira sve intrinsične i ekstrinsične parametre. Što je više slika kalibracijskog prstena uslikano iz različitih pozicija i uglova optimizacija je bolja. Nadalje, kada su koeficijenti distorzije određeni, moguće je otkloniti distorziju slika uslikanih kamerom. Nadalje, efekat distorzije može biti isključen iz daljih proračuna [3].

Pored monokularne kalibracije, moguće je vršiti i stereo kalibraciju, ukoliko su nam u aplikaciji potrebne dvije kamere ili više. Ukratko će biti objašnjena i stereo kalibracija. Obje kamere su obično pričvršćene za robota/vozilo, pa se relativna pozicija kamera ne mijenja tokom eksperimenta. Potrebno je znati transformaciju koordinata između same dvije kamere. Obično su kamere postavljene tako da postoji samo lateralna translacija među njima i ta udaljenost je uvijek poznata. Nakon toga obje se kamere kalibriraju pojedinačno da bi bili određeni intrinsični parametri. Poslije obje kamere slikaju kalibracijski prsten u isto vrijeme. Ekstrinsični parametri obje kamere prema kalibracijskom prstenu ovako mogu biti izračunati, pa se onda ti parametri koriste za izračunavanje transformacije koordinata između kamera [3].

Kada želimo odrediti lokaciju robota, obično je kamera montirana na robota. Međutim, kako je kamera pričvršćena na određenu poziciju na robotu, onda se koordinatni sistem robota ne može smatrati kao koordinatni sistem kamere. To se mora uzeti u obzir pri navigaciji robota, inače će doći do sudara robota sa preprekama, jer prepreka još nije dovoljno blizu kamere. Tada se obavlja tzv. Hand-Eye kalibracija (upravo se to odnosi na činjenicu da moramo znati šta je ruka, a šta oko robota). Dakle, ova kalibracija mjeri relativnu poziciju između koordinatnog sistema kamere i koordinatnog sistema robota. Ovdje se isto radi o transformacijama koordinata. Ukoliko tačno poznajemo gdje je kamera montirana na robotu, pozicija robota je također poznata, ukoliko je pozicija kamere prethodno estimirana. Ovako se može kontrolirati robot jednostavno koristeći poziciju kamere. Hand-Eye kalibracija se obavlja na sljedeći način: Šahovska ploča

se postavi negdje u okolini kamere i definira se koordinatni sistem okoline. Robot se postavi negdje oko šahovske ploče, tako da kamera vidi robota. Nakon toga se mjeri tačna pozicija (distanca i orijentacija) robota i njegovog koordinatnog sistema u odnosu na stvarni koordinatni sistem. Ovo mjerjenje se obavlja ručno pa se mogu pojaviti neke male greške, nekoliko centimetara ili milimetara. Poslije se više različitih pozicija robota mjeri, da bi se poboljšao rezultat kalibracije. Ukoliko su robot i šahovska ploča na istoj horizontalnoj ravnini onda se broj stepeni slobode reducira sa šest na tri, s obzirom da se rotacija obavlja samo po z osi, a translacija po x i y osi. Poslije se obavlja mjerjenje relativne pozicije kamere u odnosu na neku drugu šahovsku ploču, slikanjem šahovske ploče kamerom. Sada kada su poznate transformacije koordinata robota u odnosu na okolinu i transformacije koordinata kamere u odnosu na okolinu može se izračunati transformacija koordinata robota u odnosu na kameru ukoliko pomnožimo samo prethodne dvije, jednačina 2.9. Dakle, iz ove relacije možemo izračunati bilo koju transformaciju koja nam je potrebna [32].

$$\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}_C^R = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}_C^W \cdot \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}_W^R \quad (2.9)$$

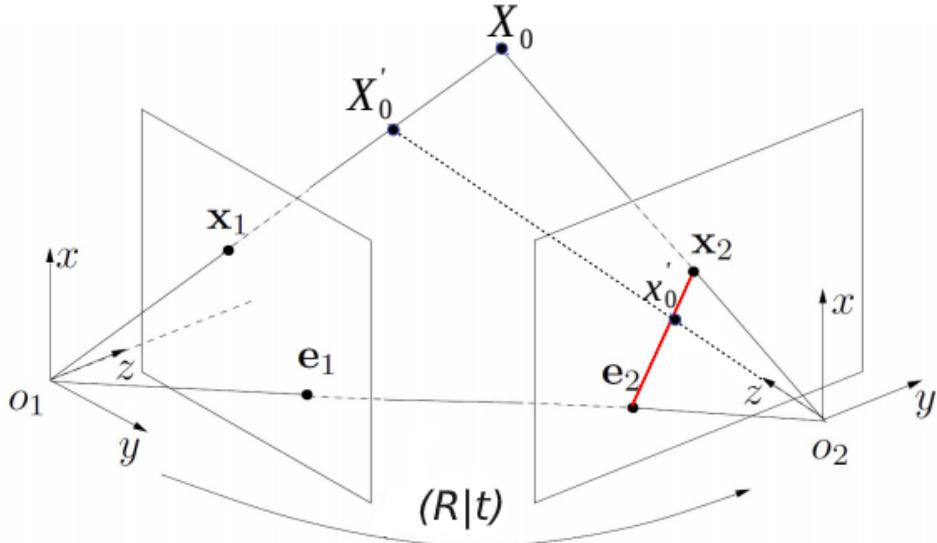
## 2.3. Geometrija dvaju pogleda

U prethodnom poglavlju je navedeno kako se vrši transformacija 3D koordinata tačke u 2D tačku na slici, tj. kako je 3D tačka projicirana na sliku. Naime, razmatran je slučaj kada se koristi jedna kamera i slika se određena scena. Lako se može primijetiti da je tada informacija o dubini scene izgubljena. Nemoguće je tada odrediti taj skalirajući faktor  $\lambda$ , jednačina 2.8. Informacija o dubini slike se može odrediti ukoliko postoji više slika te jedne scene, iz različitih uglova. Ljudske oči upravo predstavljaju stereo pogled, sa dva oka - dvije "kamere", vide 3D sliku okruženja i može se odrediti dubina te scene. Geometrija dvaju pogleda je prikazana na slici 2.4.

Ukoliko je poznata tačna distanca i orijentacija između dvije kamere, transformacije koordinata, tačke  $x_1$  i  $x_2$  pripadaju istoj stvarnoj tački  $X_o$ , čije se koordinate trebaju odrediti. Tačka  $X_o$  se može estimirati triangulacijom. U kompjuterskoj viziji postoji više načina za rješavanja triangulacije. Osnovni princip je da se pronađe presječna tačka između linija koje povezuju optičke centre kamera  $o_1$  i  $o_2$  sa tačkama  $x_1$  i  $x_2$ , te će se pronaći tako koordinate tačke  $X_o$ . U ovom slučaju kada su slike uslikane sa dvije kamere tačno je poznata transformacija koordinata između tih kamera. Može se koristiti i jedna klizna kamera koja se pomjera i slika iz različitih pozicija slike. U ovom slučaju je potrebno odrediti kako se kamera pomjerila. Da bi se dobila ova informacija koristi se epipolarna geometrija koja će biti objašnjena u nastavku [2].

### 2.3.1. Epipolarna geometrija

Epipolarna geometrija objašnjava kako su dvije slike povezane, koje je uslikala jedna kamera ili dvije približno iste kamere u okviru stereovizijskog sistema. Prvo je potrebno



Slika 2.4: Geometrija dvaju pogleda [2]

odrediti kako su tačke  $x_1$  i  $x_2$  dvaju slika, iste stvarne tačke  $X_o$ , povezane. Sljedeća jednačina prikazuje tu vezu i to se naziva epipolarnim ograničenjem.

$$x_2^T F x_1 = 0 \quad (2.10)$$

$\mathbf{F}$  je matrica  $3 \times 3$  i naziva se fundamentalna matrica. Fundamentalna matrica  $\mathbf{F}$  se može izračunati ukoliko je poznato nekoliko odgovarajućih tačaka  $x_1, i$  i  $x_2, i$ . Naprimjer, ukoliko je poznato osam ovakih tačaka, algoritam sa osam tačaka, može se odrediti matrica  $\mathbf{F}$ . S obzirom da znamo da su tačke  $x_1$  i  $x_2$  formirane množenjem matrice kamere  $\mathbf{K}$  i transformacije koordinata  $(\mathbf{R}|t)$  sa stvarnom tačkom  $X_o$ , može se pokazati da se matrica  $\mathbf{F}$  može izračunati i kao 2.11:

$$F = K^{-1}^T \cdot [t]_x \cdot R \cdot K^{-1} \quad (2.11)$$

gdje je  $[t]_x$  matrična reprezentacija vektorskog proizvoda sa matricom  $t$ . Bitna stvar koja se treba uočiti u 2.11 je da se matrica  $\mathbf{F}$  sastoji od invertirane matrice kamere  $\mathbf{K}$  i transformacije koordinata između dvije sukcesivne pozicije kamere. Nakon što se izračuna fundamentalna matrica koristeći korespondentne tačke moguće je ekstraktovati  $(\mathbf{R}|t)$  singularnom dekompozicijom. Naravno, matrica kamere mora biti poznata. Na kraju, pomjeranje kamere se može odrediti na osnovu dvije uzostopne slike. Ovo radi koristeći jednu kameru i postaje veoma korisno jer je moguće uraditi 3D triangulaciju ili čak izračunati trajektoriju kamere [33].

Još jedna bitna primjena fundamentalne matrice je što se ona može koristiti za generiranje epipolarnih linija. Na slici 2.4 postoji linija koja povezuje dva optička centra

kamera  $o_1$  i  $o_2$ . Tačke  $e_1$  i  $e_2$ , gdje se linija presjeca sa ravninom slike, se nazivaju epipolovi. Kada se tačka  $X_o$  pronađe na lijevoj slici (tačka  $x_1$ ), sljedeći korak je pronaći korespondentnu tačku  $x_2$  na desnoj slici. S obzirom da je poznata tačka  $x_1$  to znači da se tačka  $X_o$  mora nalaziti negdje na liniji koja prolazi kroz tačke  $o_1$  i  $x_1$ . Na ovoj liniji postoje mnoge moguće pozicije tačke  $X_o$  kao npr.  $X'_o$  koje bi odgovarale lijevoj slici. Ukoliko se sve ove pozicije projiciraju na desnu sliku sve one leže na liniji koja prolazi kroz epipol  $e_2$  (crvena linija na slici 2.4). Znajući ovu liniju, traženje tačke  $x_2$  može spasti na jednu dimenziju umjesto pretrage cijele slike. Ova linija  $l_2$  se naziva epipolarnom linijom i može biti jednostavno izračunata uz pomoć fundamentalne matrice  $\mathbf{F}$  2.12. Prepostavi li se da postoji homografija  $\mathbf{H}$  za koju vrijedi  $\mathbf{H} \cdot x_1 = x_2$  za svaku tačku  $x_o$  epipolarne ravnine, tada je epipolarna linija definirana tačkama  $Hx_1$  i  $e_2$  kao  $l_2 = [e_2]_x \cdot H \cdot x_1$  [33].

$$l_2 = F \cdot x_1 \quad (2.12)$$

Vec je rečeno da epipolarna geometrija prepostavlja da nema distorzija prostora kao posljedica korištenja leća u kamери. Fundamentalna matrica opisuje odnose projekcija tačaka u realnom slučaju u kom je prostor iskrivljen iz spomenutog razloga. Kamere je moguće kalibrirati tako da one zadovoljavaju gore navedeni model. Kalibracijom se dolazi do intrinskih parametara kamere i kalibracijske matrice  $\mathbf{K}$ . Jednom kada je poznata kalibracijska matrica, moguće je normalizirati koordinatni sistem slike. Normalizirane koordinate tačke  $x$  se dobiju kao  $x' = K^{-1}x$ . Vezu koju opisuje fundamentalna matrica moguce je predstaviti drugom matricom koja uzima u obzir ograničenje na model kamere. Ta matrica se naziva esencijalna matrica te se označava sa  $\mathbf{E}$ . Za normalizirane koordinate vrijedi  $x_2^T \cdot E \cdot x_1' = 0$ . Pa je moguće doći do odnosa fundamentalne i esencijalne matrice, jednačina 2.13 [33].

$$F = K_2^{-T} \cdot E \cdot K_1^{-1} \quad (2.13)$$

gdje su  $K_2$  i  $K_1$  intrinski parametri dvaju kamera, ukoliko su one različite. Esencijalna matrica se može smatrati pretečom fundamentalne matrice. Obje matrice se koriste da bi se odredila ograničenja između korespondentnih tačaka slika, esencijalna matrica se koristi ukoliko se koriste kalibrirane kamere, jer su potrebni intrinski parametri kako bi se uradila normalizacija. Esencijalna matrica sadrži informacije o rotaciji i translaciji, koje opisuju lokaciju druge kamere u odnosu na prvu kamenu. Na osnovu svih prethodnih jednačina, vidimo da se esencijalna matrica definira kao, jednačina 2.14:

$$E = R \cdot [t]_x \quad (2.14)$$

gdje je  $[t]_x$  matrična reprezentacija vektorskog množenja sa  $t$ .

Esencijalne matrice pripadaju skupu matrica u  $\mathbb{R}^{3 \times 3}$  koji se naziva esencijalni prostor  $\epsilon$ , jednačina 2.15:

$$\epsilon = \{R \cdot [t]_x | R \in SO(3), T \in \mathbb{R}^3\} \subset \mathbb{R}^{3 \times 3} \quad (2.15)$$

gdje je  $SO(3)$  specijalna ortogonalna grupa što znači da imaju svojstvo očuvanja unutarnjeg produkta. Za sve matrice  $M \in SO(n)$  također vrijedi da im je determinanta jednaka  $+1$ . Esencijalna matrica ima dvije singularne vrijednosti koje su jednake i jednu koja je nula. Ukoliko se esencijalna matrica pomnoži sa skalarom koji nije nula rezultat je esencijalna matrica koja ima ista ograničenja [33]. Ograničenja esencijalne matrice se mogu definirati i kao, jednačina 2.16:

$$\det E = 0 \quad 2EE^T E - \text{tr}(EE^T)E = 0 \quad (2.16)$$

Posljednja jednačina je matrično ograničenje, što je zapravo sistem jednačina sa devet nepoznatih, za svaki element matrice pojedinačno. Ova ograničenja se obično koriste da se esencijalna matrica odredi iz pet korespondentnih tačaka. Esencijalna matrica ima pet ili šest stepeni slobode, zavisno od toga da li je projekcijski element ili ne. Matrica rotacije  $\mathbf{R}$  i matrica translacije  $\mathbf{t}$  imaju po tri stepena slobode, šest ukupno. Ukoliko se esencijalna matrica posmatra kao projekcijski element, onda jedan stepen slobode koji je vezan za skalarno množenje može biti oduzet pa ih ima pet [33]. Ukoliko je dat set korespondentnih tačaka moguće je estimirati esencijalnu matricu koja zadovoljava sva epipolarna ograničenja za svaku tačku u setu. Međutim, ukoliko su tačke podložne šumu, što je često u praksi, nije moguće pronaći matricu koja zadovoljava sva ograničenja tačno. Zavisno od toga kako je greška koja je povezana sa svakim ograničenjem mjerena moguće je estimirati esencijalnu matricu koja optimalno zadovoljava ograničenja za dati skup tačaka. Najčešći pristup je da se odredi najmanja kvadratna greška, problem poznat kao algoritam osam tačaka. Broj stepeni slobode implicira da za procjenu  $\mathbf{E}$  nije potrebno raspolažati sa osam tačaka. Koristeći ovu činjenicu, razvijeni su algoritmi za procjenu matrice koristeći sedam, šest i konačno samo pet tačaka. Algoritam s pet tačaka ne nudi geometrijski intuitivno objašnjenje kao što je to slučaj s algoritmom za osam tačaka i ne oslanja se na rješavanje linearног sistema jednačina [34].

Kao što je i ranije navedeno moguće je odrediti matrice rotacije i translacije između dvaju koordinatnih sistema kamera. Da bi se ove matrice odredile potrebno je uraditi dekompoziciju na singularne vrijednosti esencijalne matrice (eng. *Singular-Value Decomposition* - SVD). SVD matrice je često korištena faktorizacija matrice jer je njome moguće doći do osnovnih svojstava matrice kao što su rang, doseg i nul-prostor matrice, a koristi se u mnogim linearnim algebarskim problemima poput izračuna inverza matrice te linearne procjene najmanjih kvadrata. SVD je moguće provesti za realne i kompleksne matrice. Za matricu  $M \in \mathbb{R}^{m \times n}$  ranga  $p$  vrijedi:

- $\exists U \in \mathbb{R}^{m \times p}$  čije su kolone ortonormalne.
- $\exists V \in \mathbb{R}^{n \times p}$  čije su kolone ortonormalne.
- $\exists \Sigma \in \mathbb{R}^{p \times p}, \Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_p\}$  za koje vrijedi  $\sigma_1 \leq \sigma_2 \leq \dots \leq \sigma_p$  takvi da vrijedi  $M = U\Sigma V^T$ . Matrice  $U$  i  $V$  nisu jedinstvene s obzirom na  $M$ , dok  $\Sigma$  jeste i njene dijagonalne vrijednosti su singularne vrijednosti matrice  $M$ .

Ako se matrica  $M \in \mathbb{R}^{n \times n}$  promatra kao element koji linearno mapira tačku  $x$  u tačku  $y$ , tada je moguće promatrati odnos tih dviju tačaka na sljedeći način:

- kolone matrice  $V$  su ulazni bazni vektori prostora  $\mathbb{R}^n$
- kolone matrice  $U$  su izlazni bazni vektori prostora  $\mathbb{R}^n$
- diagonalne vrijednosti matrice  $\Sigma$  su skalarni faktori kojima se mapiraju koordinate ulazne tačke  $x$  u izlaznu tačku  $y$ .

Ako se SVD primijeni na esencijalnu matricu  $E \in \mathbb{R}^{3 \times 3}$ , uvjeti su nešto stroži. Da bi matrica  $E$  bila esencijalna matrica, njena SVD  $E = U\Sigma V^T$  mora biti takva da vrijedi  $\Sigma = \text{diag}\{\sigma, \sigma, 0\}$  gdje  $\sigma \in \mathbb{R}_+$ , a  $U, V \in SO(3)$ . Kako se standardnim postupkom SVD-a ne poštuje ograničenje da matrice  $U$  i  $V$  moraju biti iz specijalne ortogonalne grupe, a njihova vrijednost nije jedinstveno određena matricom  $E$ , moguće je doći do više rješenja za problem dekompozicije esencijalne matrice. Od mogućih rješenja ispravno se pronađu eliminacijom onih rješenja u kojima se tačka ne nalazi ispred obje kamere. Dakle oblik matrice  $\Sigma$  je, jednačina 2.17, te se definira matrica  $W$ , jednačina 2.18 [34].

$$\Sigma = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (2.17)$$

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad W^{-1} = W^T \quad (2.18)$$

Sada se matrica rotacija i translacije mogu odrediti kao, jednačina 2.19 i 2.3.1..

$$[t]_x = UW\Sigma U^T \quad (2.19)$$

$$R = UW^{-1}V^T \quad (2.20)$$

## 2.4. Zaključak

U ovom poglavlju su predstavljeni osnovni koncepti vizualne odometrije. Obašnjeno je kako nastaje slika, pinhole model kamere. Detaljno su predstavljene i euklidske transformacije ili transformacije koordinata kamere u odnosu na okolinu gdje se kamera posmatra kao kruto tijelo. Objasnjeno je i šta su to instrinski i ekstrinski parametri kamere i šta utiče na njih kada se radi o stvarnom modelu kamere u odnosu na pinhole model, te kako se te smetnje otklanjavaju. Ovi pojmovi i koncepti su objasnjeni s ciljem da se na kraju opiše i definira esencijalna matrica i kako se esencijalna matrica odredi, te kako se iz nje određuje matrica rotacija i translacije kamere na osnovu značajki sa njenih uzastopnih slika. Ovaj teorijski uvod služi da bi se razumjela naredna poglavљa koja objašnjavaju algoritam vizualne odometrije korak po korak.

## Poglavlje 3.

# Monokularna vizualna odometrija

Odometrija je postupak korištenja podataka o kretanju za procjenu relativnog položaja objekta. Kod vizualne odometrije podaci korišteni za takvu procjenu su isključivo slike ili video sekvenca. Takva odometrija nije ovisna o načinu kretanja objekta, već se može upotrijebiti neovisno o pogonu vozila ili robota. Najpoznatija upotreba ove vrste procjene položaja je vjerojatno na Mars Roveru, kao što je već ranije spomenuto. Metode korištene u ovom radu su većinom prilagođene za vizualnu odometriju za cestovna vozila. Također, ovo je primjer monokularne odometrije, što implicira da je u postupku procjene kretanja korištena samo jedna kamera koja je bila montirana na prednjem dijelu vozila. Algoritam je ukratko opisan u narednim koracima:

1. Prikupljanje slijeda ulaznih slika iz videa.
2. Vađenje (ekstrakcija) značajki sa slike.
3. Korespondencija značajki između dvaju ili više slika.
4. Estimacija poze kamere: procjena esencijalne matrice, odstranjivanje vanpopulacijskih značajki (eng. *outliers*), zatim procjena ukupne matrice rotacije i translacije u svakom okviru (eng. *frame*).

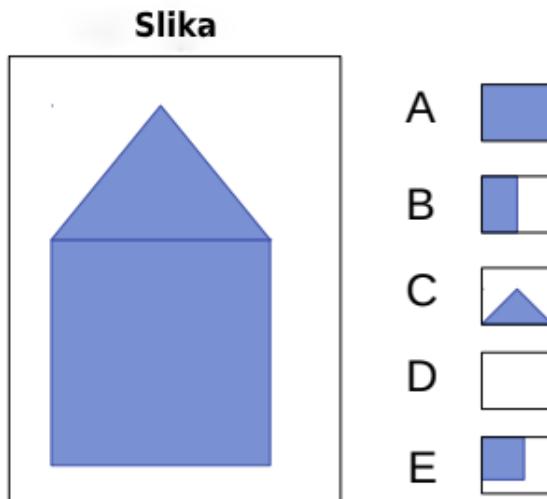
Svi prethodno navedeni koraci algoritma bit će pojašnjeni u nastavku i kako su oni realizovani.

### 3.1. Prikupljanje slijeda ulaznih slika iz videa

Video je tehnologija elektronskog snimanja, procesiranja, memorisanja, prenosa i rekonstrukcije sekvence mirnih slika koje predstavljaju scenu u pokretu. Video tehnologija je prvenstveno bila razvijena za televizijske sisteme, ali je kasnije razvijen niz formata koji čine da video bude dostupan širem krugu korisnika. Standardi na kojima se doskora zasnivala televizija i sam video postavljeni su prije skoro pola vijeka i promjene u ovoj oblasti su spore. U novije vrijeme pojavljuju se brojni standardi koji se trude da prevaziđu postojeća ograničenja videa i da doživljaj gledanja podignu na viši nivo. Svi oni se kreću u pravcu digitalnog procesiranja slike. Video je niz slika kako je ranije navedeno, slike se objavljuju određenom brzinom i nazivamo ih frejmovi (eng. *frames*) videa. Naime, procesiranje videa zasniva se na digitalnoj obradi svakog pojedinačnog frejma, slike.

### 3.2. Vađenje (ekstrakcija) značajki sa slike

U vizualnoj odometriji dva ili više uzastopnih frejmova se koristi za izračunavanje kretanja kamere. Da bi se ovo uradilo, potrebno je saznati kako se pozicija projiciranih 3D tačaka mijenja kroz slike. Potrebno je pronaći određene tačke na slici i provjeriti kako su se one pomjerile na narednoj slici. Ove tačke moraju biti jedinstvene kako bi ih kompjuter mogao pronaći. Zato se ove tačke nazivaju značajke ili ključne tačke (eng. *features and keypoints*). Postoje mnogi problemi pri ovoj ekstrakciji, na slici 3.1 je prikazan jedan takav problem, tzv. problem otvora (eng. *aperture problem*).



Slika 3.1: Aperture problem [4]

Ova slika prikazuje jednostavnu kuću, pretpostavimo da su ovi dijelovi A, B, C, D, E pronađeni na drugoj slici iste scene. Naime, potrebno je sada pronaći ove isječke na originalnoj slici. Isječci A i D prikazuju homogenu površinu slike i nemoguće je pronaći tačno odakle su oni isječeni. Pa ove "značajke" je jako teško pronaći, jer nisu jedinstvene i moguće je pronaći bezbroj rješenja. Samo, zapravo, isječci C i E koji pretstavljaju uglove kuće mogu biti eksplicitno pronađeni na originalnoj slici. Njihovu poziciju je moguće egzaktno pronaći. Pa vidimo da su uglovi zapravo dobre značajke za pronalaženje u kompjuterskoj viziji.

Postoje različiti pristupi pronalaska robustnih značajki na slici. Postoje prvo detektori kornera/uglova i tzv. blob detektori [4]. Blobovi su zapravo dijelovi na slici koji se razlikuju po svojstvima od svog okruženja. Ukoliko se želi odabrati odgovarajući detektor za neku aplikaciju potrebno je uzeti u razmatranje sljedeće [4]:

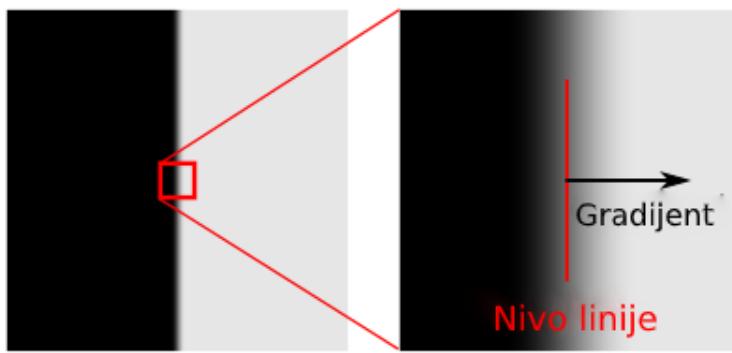
- Robusnost (invarijantnost rotacija, translacije, skaliranja i iluminacije).
- Tačnost pri lokalizaciji značajke.
- Trošak izračunavanja algoritma.

Postoji mnogo različitih algoritama za detekciju značajki. U ovom radu bit će korišteni blob detektori. Naime, kako je cilj realizovati algoritam za cestovna vozila, najčešće

značajke koje se pojavljuju tada na slikama su zapravo linije različitih dužina. Linije su uočljive kao rubovi same ceste, bijele linije na cesti koje razdvajaju kolovozne trake, zatim linije na okolnim zgradama, objektima, autima, te ih je najlakše pratiti u ovom slučaju. Detektori koji će biti korišteni je **LSD** (eng. *Line Segment Detector*), **PHLT** (eng. *Probabilistic Hough Line Transform*) i **SHLT** (eng. *Standard Hough Line Transform*) i bit će objašnjeni u nastavku.

### 3.2.1. Line Segment Detector

LSD (eng. *Line Segment Detector*) detektor je vremenski linearan detektor koji daje veoma precizne rezultate. Dizajniran je tako da može biti upotrebljen na svakoj digitalnoj slici. Ovaj detektor može sam da kontrolira lažne detekcije. Cilj je detektovati lokalno ravne konture na slici, koji se nazivaju linijskim segmentima. Zone gdje se nalaze konture na slici su zapravo promjene gdje se sivi dijelovi brzo mijenjaju sa tamne na svijetlu boju i obrnuto. Naime, gradijent i te linije promjene na slici su najvažniji koncept i ilustrirane su na slici 3.2 [5].

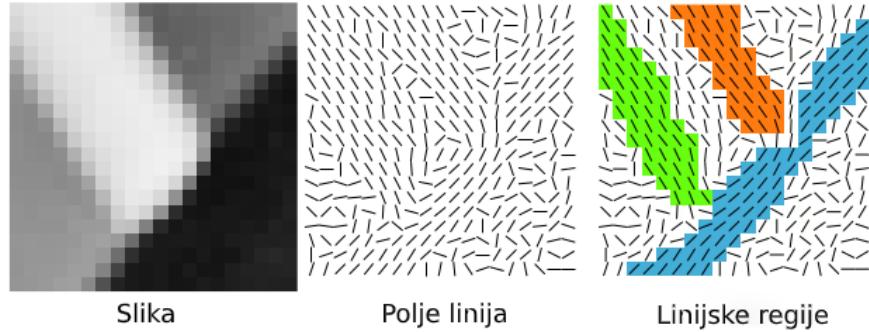


Slika 3.2: Gradijent i linije promjene [5]

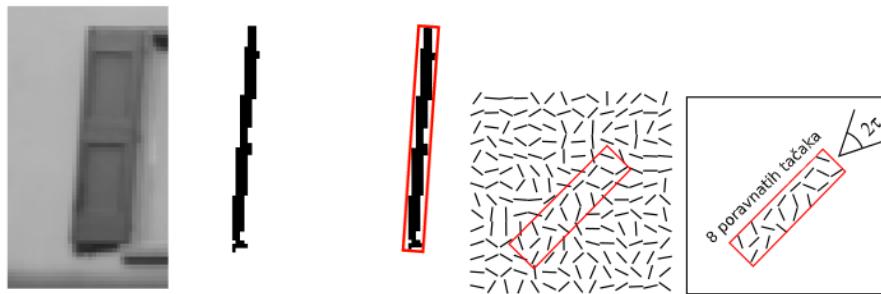
Algoritam počinje tako što proračunava ugao linije promjene ili nivo linije (eng. *level-line*) za svaki piksel da bi kreirala polje linija promjene (eng. *level-line field*), tj. polje jediničnih vektora takvo da su svi vektori tangente na liniju promjene koja prolazi kroz njihovu početnu tačku. Nadalje, ovo polje je podijeljeno na povezane dijelove piksela koji imaju zajednički ugao linije promjene za neku toleranciju  $\tau$ . Ovi povezani dijelovi piksela nazivaju se linijske regije/dijelovi (eng. *line-support regions*), slika 3.3.

Svaki linijski dio (grupa piksela) je potencijalni kandidat za linijski segment. Može se uočiti da su ovi segmenti i grupe piksela pravougaonog oblika. Odabere se pravougaonik koji će da okruži cijelu linijsku grupu piksela, slika 3.4. Svaki pravougaonik se koristi za validaciju. Pikseli u pravougaoniku čiji ugao linije promjene odgovara uglu pravougaonika za neku toleranciju  $\tau$  nazivaju se poravnatim tačkama, slika 3.4. Ukupan broj piksela u pravougaoniku  $n$  i broj poravnatih tačaka  $k$  se koristi da se validira da li je pravougaonik linijski segment. Ovdje je ukratko objašnjen LSD detektor i njegov osnovni princip rada [5].

Pri proračunu gradijenta na slici broj operacija je proporcionalan broju piksela slike. Pikseli su grupirani i operacija je obavljena u linearном vremenu. LSD algoritam, na kraju, tj. njegovo vrijeme izvršavanja je proporcionalno broju piksela na slici. Bitno



**Slika 3.3:** Linijski dijelovi [5]



**Slika 3.4:** Pravougaona aproksimacija linijskog regiona i poravnate tačke [5]

je spomenuti da LSD algoritam koristi grayscale slike. Grayscale slika je formirana od različitih nijansi sive gdje svaki piksel slike sadrži vrijednost koja definira njegovu nijansu sive, ako je slika osmobilna onda je to vrijednost od 0-255 [5]. Finalni rezultat LSD detektora moguće je vidjeti na slici 3.5.



**Slika 3.5:** Primjer detekcije linija pomoću LSD detektora [6]

### 3.2.2. Hough Line Transform

U slučaju kada je potrebno detektovati neki objekat na slici važno je smanjiti broj podataka ali i dalje zadržati bitne karakteristike i strukturne informacije. Detektovanje ivica moguće je značajno smanjiti broj podataka, ali izlaz ovog detektora je i dalje slika koja je opisana pikselima. Ukoliko se definiraju linije koje mogu biti opisane jednačinom, broj podataka se može značajno smanjiti. Za detekciju linije definira se i Houghova transformacija, mada postoji Houghova transformacija i za detekciju drugih oblika, krugova, elipsi itd.

Houghova transformacija je dobila ime po Paulu Houghu koji je patentirao ovaj metod 1962. godine. Vrši se transformacija između Kartezijevog prostora i parametarskog prostora gdje se može definirati prava linija. Linije se predstavljaju u Houghovom prostoru. Linije mogu biti jedinstveno opisane pomoću dva parametra. Obično se koristi jednačina 3.1 sa parametrima  $a$  i  $b$  [7].

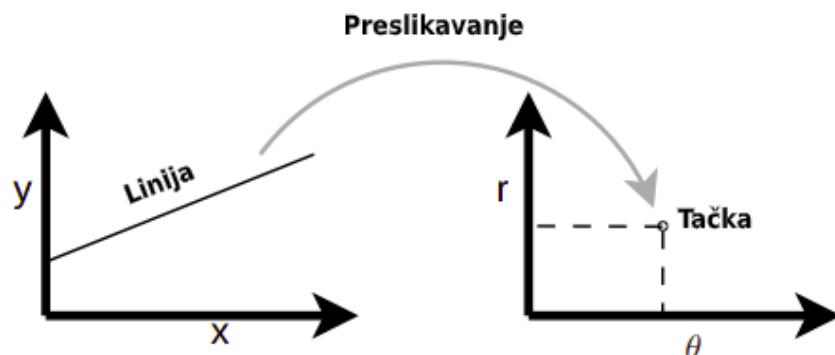
$$y = a \cdot x + b \quad (3.1)$$

Međutim ova forma nije pogodna za predstavljanje vertikalnih linija. Pa Houghova transformacija koristi oblik kakav je predstavljen u jednačini 3.2 tj. jednačinom 3.3 kako bi bilo slično jednačini 3.1. Parametri  $\theta$  i  $r$  su ugao linije i udaljenost linije od koordinatnog početka [7].

$$r = x \cdot \cos(\theta) + y \cdot \sin(\theta) \quad (3.2)$$

$$y = -\frac{\cos(\theta)}{\sin(\theta)} + \frac{r}{\sin(\theta)} \quad (3.3)$$

Sve linije mogu biti predstavljanje u ovoj formi gdje je  $\theta \in [0, 180]$  ili  $\theta \in [0, 360]$  i  $r \geq 0$ . Dakle, Houghov prostor linija ima dvije dimenzije  $\theta$  i  $r$ , te je linija predstavljena jednom tačkom, jedinstvenim setom parametara  $(\theta_o, r_o)$ . Preslikavanje linija-tačka je prikazano na slici 3.6 [7].



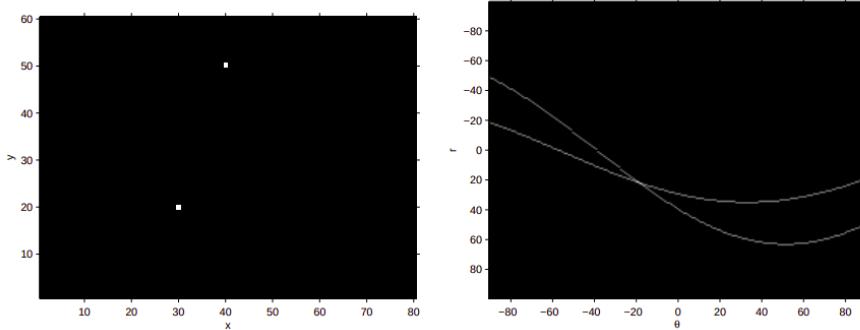
**Slika 3.6:** Preslikavanje linije u Houghov prostor [7]

Važan koncept Houghove transformacije je preslikavanje pojedinačnih tačaka. Ideja je da se tačka preslika na sve linije koje mogu proći kroz tu tačku.

Houghov algoritam za detekciju pravih linija se može predstviti sljedećim koracima [7]:

1. Detekcija ivica korištenjem Canny Edge detektora.
2. Preslikavanje tačaka sa ivica u Houghov prostor i pohranjivanje u akumulator.
3. Interpretacija akumulatora i za određivanje linija beskonačne dužine. Ova interpretacija se uradi tako što se postavi određeni prag ili ograničenje.
4. Pretvaranje beskonačnih linija u linije konačne dužine.

Linije konačne dužine mogu biti superponirane na orginalnu sliku. Houghova transformacija uzima binarnu sliku sa detektovanim ivicama kao ulaz i pokušava da pronađe ivice koje su detektovane kao prave linije. Svaka tačka ivice sa binarne slike sa ivicama se preslikava na sve moguće linije koje mogu proći tom tačkom. Binarna slika sa ivicama ima mnogo tačaka, svaka tačka je transformirana u liniju u Houghovom prostoru, pa područja gdje se najviše linija u Houghovom prostoru presjeca se interpretiraju kao prave linije. Ovaj princip je prikazan na slici 3.7 [7].



**Slika 3.7:** Transformacija dvije tačke  $p_o$  i  $p_1$  (lijeva slika) u dvije linije u Hough-vom prostoru (desna slika). Presijecanje dvije linije u Houghovom prostoru ukazuje na liniju koja prolazi kroz obje tačke  $p_o$  i  $p_1$  [7]

Da bi se odredila područja gdje se većina Houghovim linija presijeca koristi se akumulator koji pokriva cijeli Houghov prostor. Broj dimenzija akumulatora odgovara broju nepoznatih parametara u Houghovom transformacijskom problemu, dakle za detekciju linije, nepoznata su dva parametra. Akumulator interpretira linije kao beskonačne kada se sve tačke ivica transformiraju. Najosnovniji način da se detektuje linija je da se postavi određeni prag za akumulator i da se sve vrijednosti iznad praga definiraju kao linije.

Iz svega navedenog se vidi da Houghova transformacija detektuje linije sa datim parametrima  $r$  i  $\theta$ , te nema informacije o dužini linije. Sve detektovane linije su beskonačne dužine. Ukoliko želimo detektovati linije konačne dužine onda se mora provesti detaljnija analiza područja sa ivicama na binarnoj slici. Za ovaj pristup koristi se tzv.

**Probabilistic Hough Line Transform - PHLT** [7].



Slika 3.8: Primjer detekcije linija SHLT (lijeva slika) i PHLT (desna slika) [8]

Houghova transformacija nije brz algoritam gdje se detektuju linije beskonačnih dužina na nekoj slici. S obzirom da je potrebna dodatna analiza da bi se detektovale konačne linije, algoritam postaje još sporiji. Način da se ubrza standardna Houghova transformacija (SHLT) i da se pronađu linije konačne dužine koristi se PHLT. Osnovna ideja ovog pristupa je da se slučajno odabereni pikseli, sa binarne slike sa ivicama, transformiraju u akumulator. U slučaju da tzv. bin u akumulatoru koji odgovara određenoj beskonačnoj liniji ima vrijednost iznad postavljenog praga, slika sa ivicama duž te linije se pretražuje da bi se provjerilo da li postoje duž te linije jedna ili više konačnih linija. Onda se svi pikseli na toj liniji se obrišu sa slike sa ivicama. Na ovaj način algoritam vrati konačne linije. PHLT je brži algoritam i zahtjeva manje memorije od SHLT algoritma [7]. Primjer detekcije linija koristeći SHLT i PHLT je prikazan na slici 3.8.

U ovom radu korišteni su ovi blob detektori, kao što je već i ranije navedeno, na način da se nakon detektovanja linija na frejmovima za nastavak algoritma koriste pojedine tačke linije. Uzeto je da to budu krajnje tačke linije, tj. njihove koordinate. Algoritam ne uzima u obzir detektovane linije vrlo malih dužina. Prethodna "ograničenja" su implementirana u algoritmu s ciljem smanjenja vremena izračunavanja algoritma, memorija itd. Dakle, krajnje tačke linija su značajke (eng. *features or keypoints*).

### 3.3. Korespondencija značajki

Prethodno poglavje govori o ekstrakciji značajki sa slike. Postoje različite tehnike za ekstrakciju značajki sa slike, kao što je i navedeno u prethodnom potpoglavlju. Svaki algoritam ima prednosti i mane. Razlog zašto je potrebno naći ove jedinstvene i karakteristične značajke je zato što se žele pronaći tačke na različitim slikama koje su uslikane sa raznoraznih pozicija. Naime, te pronađene značajke se nastoje pronaći na dvije ili više slike. Postoje dva različita pristupa pri rješavanju ovog problema: praćenje značajki u slijedu slika (eng. *feature tracking*) i pronalaženje (može se reći uparivanje) značajki na slikama (eng. *feature matching*).

Praćenje značajki u slijedu slika zapravo znači da je kretanje značajke slijedeno kroz niz slika. Prvo je potrebno detektovati značajke na početnoj slici, kako je to objašnjeno

u prethodnom potpoglavlju. Pretpostavljeno je da se ove značajke ne pomjeraju mnogo na sljedećoj slici i da njihov izgled ostaje isti. Nakon određenih izračunavanja moguće je pratiti značajku koja je jednom pronađena od jedne do sljedeće slike. Jedan od najvažnijih pristupa pri rješavanju ovog problema praćenja značajki je KLT (eng. *Kanade-Lucas-Tomasi Tracker*) [35]. Prema pretpostavci da se dvije slike ne mijenjaju mnogo, ova tehnika je posebno korisna kada se radi o slikama koje su uslikane uzastopno, jedna za drugom, sa jedne kamere. Dakle, kamera se ne pomjera mnogo između dvije slike. Primjer praćenja značajki je prikazan na slici 3.9.

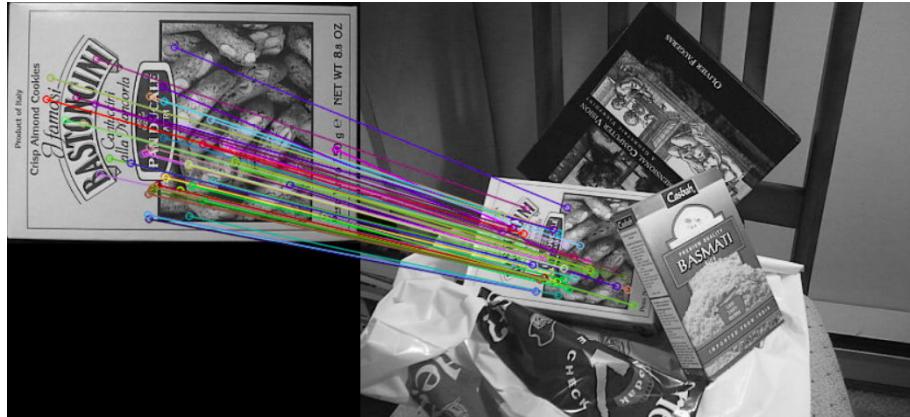


Slika 3.9: Primjer praćenja značajki pomoću Kanade-Lucas-Tomasi metode [9]

Ovaj rad je implementacija monokularne vizualne odometrije, pa je za praćenje značajki u slijedu slika jedne kamere korišten upravo prethodno spomenuti metod.

U slučaju stereo vizualne odometrije prisutne su dvije kamere i istovremeno su uslikane dvije slike, lijeva i desna. Potrebno je pronaći i upariti tačke sa lijeve i desne slike. U ovom slučaju udaljenost među kamerama nije tako mala, pa se tačke, koje se nalaze vrlo blizu ispred stereo kamere, mogu pojaviti na dosta različitim pozicijama na lijevoj i desnoj slici. U ovom slučaju koristi se pristup pronalaženja/uparivanja značajki na slikama (eng. *feature matching*). Pronalaženje/uparivanje značajki na slikama odvija se na sljedeći način: Prvo se detektuju značajke na lijevoj i desnoj slici pojedinačno. Pa je određena površina oko značajke ekstraktovana da bi opisala tu značajku. Ova površina koja se sastoji od piksela se naziva deskriptor (eng. *descriptor*) značajke. Ovaj proces se uradi za svaku značajku pa imamo niz deskriptora koji odgovara jednoj od značajki. Nadalje, deskriptori se upoređuju prolazeći kroz svaku sliku s ciljem pronašlaska sličnih značajki.

Postoje različiti algoritmi i tehnike za ovaj pristup kao što je Brute-Force-Matcher sa različitim deskriptorima (ORB, SIFT, FAST itd.) ili Flann-Matcher. Primjer uparivanja značajki na dvije slike pomoću Flann-Matcher tehnike je prikazano na slici 3.10.



**Slika 3.10:** Primjer uparivanja značajki na dvije slike pomoću Flann-Matcher tehnike [10]

### 3.3.1. Kanade-Lucas-Tomasi (KLT) metod praćenja značajki

Naime, kako je prethodno rečeno, ovaj rad se bavi problemom monokularne vizualne odometrije pa su podaci na osnovu kojih je ovaj algoritam implementiran zapravo slijed slika sa jedne kamere - video sekvenca. Ove slike su uslikane velikom brzinom (npr. 10 frejmova po sekundi), prizor se neće puno promijeniti među uzastopnim slikama, iako se kamera pomjera. Dakle, nema velikih promjena u rotaciji, skaliranju i iluminaciji pa nije potreban neki robustni "feature matcher".

Kanade-Lucas-Tomasi Tracker je predstavljen 1981. godine i još uvijek se koristi i jako je bitan metod praćenja značajki. U ovom dijelu će biti objašnjene osnovne ideje i koncept ove metode.

Na početku, postoje tri ključne pretpostavke za svojstva dvaju uzastopno uslikanih slika:

1. Postoji malo pomjeranje, pa se ni tačke ne pomjeraju mnogo.
2. Osvjetljenost je konstantna, pa tačke izgledaju isto na svakom frejmu (imaju isti intenzitet).
3. Postoji prostorna koherencija, pa tačke koje su blizu imaju isto kretanje.

Kada je u pitanju video sekvenca sve ove pretpostavke vrijede za frejmove tog videa. Sljedeći korak je razmatranje kako se značajka pomjera sa jedne slike na drugu.

Prvo je tačka slike na poziciji  $(x, y)$ . Dalje se tačka pomjeri za  $(u, v)$  pa će u narednom frejmu imati poziciju  $(x+u, y+v)$ . Pa prema pretpostavci se dobije jednačina jednakosti osvjetljenja, 3.4:

$$I(x, y, t) = I(x + u, y + v, t + 1) \quad (3.4)$$

gdje je  $I(x, y, t)$  intenzitet tačke  $(x, y)$  u trenutku  $t$ . Sada je na desnu stanicu primjenjen Taylorov razvoj da bi jednačina bila linearizirana. Nakon toga ubaci se jednačina jednakosti

osvijetljenja pa imamo, 3.5:

$$\begin{aligned} I(x+u, y+v, t+1) &\approx I(x, y, t) + I_x u + I_y v + I_t \\ \Leftrightarrow I(x+u, y+v, t+1) - I(x, y, t) &\approx I_x u + I_y v + I_t \\ \Rightarrow 0 &\approx I_x u + I_y v + I_t \end{aligned} \quad (3.5)$$

Da bi tačka bila praćena, zapravo cilj je estimirati pomjeranje  $(u, v)$  na osnovu jednačine 3.5. Problem je što je poznata samo jedna jednačina, a dvije nepoznate  $u$  i  $v$ . Ovdje je potrebno iskoristiti treću pretpostavku. Pretpostavlja se da svi susjedni pikseli imaju isto pomjeranje  $(u, v)$ . Naprimjer, ukoliko se uzme  $3 \times 3$  isječak susjednih piksela oko tačke dolazi se do sljedeće jednačine 3.6 gdje su  $p_1, p_2, \dots, p_9$  susjedni pikseli.

$$\begin{pmatrix} I_x(p_1) & I_y(p_1) \\ \dots & \dots \\ I_x(p_9) & I_y(p_9) \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} I_t(p_1) \\ \dots \\ I_t(p_9) \end{pmatrix} \quad (3.6)$$

Ovaj sistem iz prethodne jednačine 3.6 se može koristiti da se odredi  $u$  i  $v$  koji zadovoljavaju ovu jednačinu. Nadalje, nova lokacija tačke  $(x+u, y+v)$  može biti estimirana. Naime, omogućeno je praćenje određene tačke kroz niz slika. Za ovu aplikaciju ovaj pristup je odličan, pa je upravo ovaj KLT metod praćenja korišten u ovom radu.

### 3.4. Estimacija pozicije kamere

Glavni cilj ove teze je implementacija algoritma za lokalizaciju kamere korištenjem prikupljenih vizualnih podataka - slika. U prethodnom poglavlju je objašnjeno kako se detektuju karakteristične značajke na slikama, te kako se određuje njihova pozicija na drugim slikama. Sljedeći korak je određivanje transformacije koordinata kamere gdje se kamera posmatra kao kruto tijelo -  $(R|t)$ . Ovaj proces estimacije kretanja kamere na osnovu videa se naziva vizualna odometrija (eng. *Visual Odometry* - VO). Davide Scaramuzza je objavio sveobuhvatni pregled vizualne odometrije općenito i neke različite metode implementacije [13]. Postoje uglavnom tri pristupa prema ovom naučnom radu. Postoje 2D-2D, 3D-3D i 3D-2D tehnike koje mogu biti korištene. Sve ove tehnike bit će detaljno objašnjene u nastavku.

Najstariji pristup je monokularni 2D-2D pristup. Ovaj naziv je izabran zato što su 2D tačke slike upoređuju sa drugim 2D tačkama slike. Opći algoritam radi na sljedeći način: Prvo je slika uslikana i detektovane su značajke. Zatim je uslikana sljedeća slika (istom kamerom) i značajke su ponovo detektovane. Ovako su obezbijeđena dva skupa značajki koji se mogu korespondirati. Nakon procesa korespondencije, esencijalna matrica dvije slike može biti izračunata. Nadalje ta esencijalna matrica je dekompozirana kako bi se odredile transformacije krutog tijela između dvije uzostopne slike. Ovaj metod je izravan i poprilično robusan. Još jedna prednost ovog metoda je što je potrebna

samo jedna kamera. Međutim, mana ovog pristupa je tzv. dvosmislenost skalirajućeg faktora. To znači da je matrica rotacije kamere potpuno određena, ali estimirana matrica translacije ovisi o nepoznatom skalirajućem faktoru. Naprimjer, može biti izračunato da se kamera pomjerila za pet jedinica po x-osi, ali relacija između jedne jedinice i metra je nepoznata. Postoje načini kako se ovo može riješiti. Kako je to riješeno u ovom radu bit će objašnjeno u nastavku.

Sljedeći pristup je tzv. 3D-3D pristup. Iz naziva se može zaključiti da se upoređuju dva seta 3D tačaka kako bi se estimirala pozicija kamere. Da bi se doble 3D tačke sa 2D slike, koristi se stereo kamera. Ponovo, značajke se detektuju i koresponduju između lijeve i desne kamere. Nakon stereo kalibracije poznate su pozicije kamera, jedne u odnosu na drugu. Sa ovom informacijom dalje moguće je provesti tzv. proces triangulacije da bi se doble 3D koordinatne detektovanih značajki. Nakon proračunavanja 3D tačaka, pozicije kamera su određene u metrima. Nakon ovoga proračuna dvije kamere ponovo uslikaju dvije nove slike. Ovaj novi skup 3D tačaka slika je ponovo trianguliran, pa je poznat novi skup 3D tačaka. Usporedbom ova dva skupa tačaka, moguće je sada estimirati kretanje stereo kamere. Pozicija stereo kamere je estimirana u metrima. U stvarnosti uvijek postoji neki šum na slici i netačnosti, pa proces triangulacije nije upotpunosti tačan i unosi grešku. Proces triangulacije se obavlja dva puta da bi se odredila pozicija, pa je greška dodana dva puta.

Posljednji pristup koji će biti predstavljen ovdje je 3D-2D metod [14]. Ovaj metod sličan je 3D-3D metodu, 3D tačke su triangulirane stereo kamerom u prvom koraku. U narednom trenutku uzima se u obzir samo slika jedne kamere. Tačke su detektovane i korespondiraju se sa prethodno trianguliranim 3D značajkama. Transformacije krutog tijela su izračunate poređenjem 3D i 2D skupa tačaka. Ovaj metod se odnosi na tzv. problem perspektive u tački (eng. *perspective in point* - PnP problem). Prednost ove metode je što je greška triangulacije prisutna samo jednom.

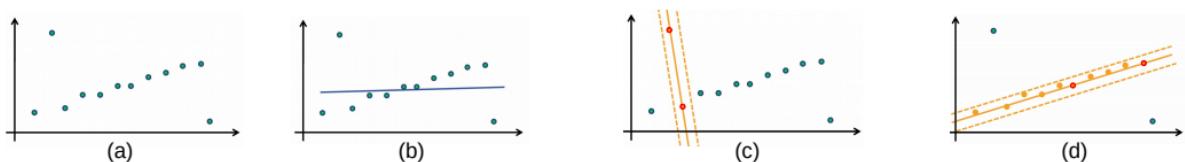
Tema ovog rada je prvi pristup pa će u samoj implementaciji biti objašnjeno na osnovu vizualnih podataka koji su korišteni kako je problem skaliranja riješen.

### 3.4.1. Odstranjivanje vanpopulacijskih značajki

Pri izračunavanju kretanja kamere zbog šuma na slikama i netačnosti pri mjeranjima, nemoguće je pronaći transformaciju ( $R|t$ ) koja savršeno odgovara svim tačkama u skupu. Zapravo, potrebno je pronaći transformaciju koja minimizira reprojekcijsku grešku svim tačkama u skupu. Algoritam koji prati značajke kroz dvije uzastopne slike će ponekad napraviti grešku, postojat će nepogodne korespondencije među pojedinim tačkama. Važno je znati kako otkloniti ove nepogodne korespondencije - vanpopulacijske značajke (eng. *outliers*). Fischler i Bolles su predstavili metod RANSAC (eng. *Random Simple Consensus*) za odstranjivanje vanpopulacijskih značajki u skupu [36]. U ovom radu je korišten kako bi ovaj VO algoritam bio robusniji u odnosu na outliere i druge nepogodne značajke. Ovdje će biti objašnjen ovaj algoritam.

RANSAC je iterativni metod za izračunavanje parametara modela koristeći promatrane podatke koji sadrže neke vanpopulacijske vrijednosti. Postavljaju se unaprijed dva

uslova: Prvi, postojo mnogo dobrih tačaka (eng. *inliers*) u podacima koji će odgovarati stvarnom modelu. Drugi, zašumljene ili vanpopulacijske tačke su raspoređene nasumično. Ukoliko vrijede ove dva uslova, RANSAC ekstraktovati skup tačaka koji sadrži samo dobre tačke (eng. *inliers*) originalnih podataka. Naime, odabere se nasumično podskup tačaka i pretpostavi se da su sve tačke inlieri. Na osnovu ovog podskupa kreira se model koji se verificira tako što se upoređuje sa ostalim tačkama [11]. Naredna slika pokazuje jednostavni primjer koji ilustrira ovaj pristup, slika 3.11.



Slika 3.11: Primjer RANSAC metode [11]

U primjeru 3.11(a) prikazan je skup 2D tačaka. U ovom modelu konkretno svi inlieri bi trebali ležati na pravoj liniji. Većina tačaka odgovara ovom modelu, postoje samo dva outliera. U primjeru 3.11(b) postoji linija koja minimizira kvadratnu grešku između linije i tačaka. Očigledno je da ova linija ne odgovara većini tačaka, zbog dva outliera. Može se zaključiti da je bitno da se otklonu svi outlieri prije korištenja nekog optimizacijskog algoritma. U primjeru 3.11(c) prikazana je jedna iteracija RANSAC-a. Dvije tačke iz podataka su odabrane nasumično i korištene su za estimaciju linije. Nakon toga je provjereno koliko tačaka iz skupa leži otprilike na ovoj liniji sa određenom netačnosti. U primjeru 3.11(c) samo su dvije tačke (dvije odabrane tačke) blizu linije. U primjeru 3.11(d) ponovo su odabrane dvije nove tačke nasumično. Sada je većina tačaka inlieri. Pa ova nova linija daje dobar model za date podatke [11].

Dakle, umjesto da se iz svih podataka računa model i onda uklanjuju tačke koje ga ne zadovoljavaju, RANSAC uzima najmanji broj značajki potreban za procjenu modela te računa koliko je postojećih značajki moguće uklopiti u izračunati model. Da bi se moglo izračunati pripada li određena značajka modelu ili ne, potrebno je postaviti prag (udaljenost) unutar kojeg je značajka dio modela, a izvan kojeg se može klasificirati kao nepogodna. Taj prag se označava sa  $t$  (eng. *threshold*) i potrebno ga je eksperimentalno utvrditi. RANSAC će odrediti broj značajki koje odgovaraju početnom modelu koji se nalaze unutar praga. Ako je pronađen model s dovoljnim brojem značajki, testira se njegova kvaliteta (na svim značajkama). Ako je bolji od bilo kojeg dotad pronađenog modela, pamti se kao najbolji. Nakon zadanoj broj iteracija vraća se najbolji model. Sa ovim metodom moguće je ukloniti sve outliere iz skupa značajki i dobiti robusnu estimaciju kretanja kamere ( $R|t$ ). Problem je što će veliki broj iteracija povećati vrijeme izračunavanja. Broj iteracija  $N$ , koji je potreban da bi se dobio dobar rezultat, zavisi od nekoliko faktora. Najvažniji je odnos broja inliera i outliera u skupu podataka. U slučaju kada postoji mnogo outliera postaje manje vjerovatno da je moguće napraviti neki nasumični odabir tačaka tako da budu svi inlieri. Vjerovatnoća odabira tako "dobrog" skupa podataka je (jednačina 3.7):

$$\omega = \frac{\text{broj inlier-a}}{\text{ukupan broj tačaka}}, \quad 0 < \omega \leq 1 \quad (3.7)$$

Kada je potreban minimalan broj s tačaka da bi se izvršio određeni algoritam, vjerovatnoća odabira s tačaka uzastopno koje su sve inlieri je  $\omega^s$ . Ukoliko RANSAC treba da pronađe takav skup "dobrih" tačaka sa vjerovatnoćom p, potrebni broj itrecija N, (jednačina 3.8):

$$N = \frac{\log(1 - p)}{\log(1 - \omega^s)} \quad (3.8)$$

U većini slučajeva p se odabere tako da bude veliko (naprimjer 0.999) da sigurno ne bi bilo outliera u optimalnom skupu tačaka. Dakle, dva faktora odnos  $\omega$  i minimalan broj tačaka podskupa s najviše određuju broj iteracija algoritma N. Da bi ovaj broj N bio što niži,  $\omega$  mora biti što veće (blizu 1) dok je broj s minimiziran. Da bi  $\omega$  imala veliku vrijednost ne smije biti mnogo nepogodnih korespondencija među značajkama sa slikom. Zato je potrebno odabrati vrlo tačne algoritam detekcije i praćenja značajki [11].

Mogući parametri za ovaj algoritam su broj iteracija, prag tolerancije i broj tačaka koje moraju uklapati u model da bi on bio valjan. U ovom radu RANSAC metoda i procjena esencijalne matrice se obavlaju istovremeno (uklopljeni su u istu funkciju). RANSAC metoda za uklanjanje vanpopulacijskih značajki je robusnija i daje bolje rezultate od metode najmanjih kvadrata koju se također može koristiti za ovaj postupak, razlog tome je što podaci mogu biti toliko podložni šumu i time neispravni da je nemoguće iz njih konstruirati ispravan model.

### 3.4.2. Procjena matrice rotacije i translacije

Posljednji korak vizualne odometrije je estimacija ukupne pozicije kamere/roboata. Računa se kretanje robota između početnog i zadnjeg frejma. Da bi se lokalizirao robot, potrebno je odrediti cijelu trajektoriju kretanja od početka. Početna pozicija robota se uzima kao referentni koordinatni sistem. Ukupna pozicija robota se onda izračuna jednostavno nadovezivanjem trenutnog kretanja i ukupne pozicije kao u jednačinama 3.9 i 3.10.

$$R_{total} = R_{trenutno} \cdot R_{total} \quad (3.9)$$

$$t_{total} = t_{total} + scale \cdot R_{total} \cdot t_{trenutno} \quad (3.10)$$

Ukupna pozicija se računa za svaki vremenski trenutak, kada algoritam dobije novu sliku iz video sekvence. Informacija o faktoru skaliranja za translaciju se dobije iz drugog izvora i to će biti opisano u narednom poglavlju ovog dokumenta.

Poslije svih ovih koraka algoritam vizualne odometrije je završen. Međutim, postoji problem pri računanju ukupne pozicije nadovezivanjem svih prethodnih kretanja. Algoritam

radi vrlo brzo jer se koristi video sekvenca, trenutni prijenos ili već snimljena video sekvenca, pa se i ukupna pozicija ažurira jako brzo. Postoji u startu mala greška pri estimiranju pozicije, ali se ova greška nakuplja vremenom. Naprimjer, ako je brzina (eng. *frame-rate*) video sekvence 10 frejmova po sekundi i ako je translacijska greška 1 mm, poslije 10 sekundi video sekvence translacijska greška će biti 100 mm ( $1\text{mm} \cdot 10\text{fps} \cdot 10\text{s}$ ). Stvara se određeni drift između estimirane pozicije i stvarne pozicije, što je generalno problem odometrije. Greška se propagira tokom vremena. Algoritam ima bolju tačnost na početku ali ona se vremenom smanjuje. Postoje mnogi načini kako se ovaj drift može smanjiti, kako su te metode dosta komplikovane one nisu implementirane u ovom radu.

### 3.5. Zaključak

U ovom poglavlju je detaljno objašnjen algoritam monokularne vizualne odometrije. Objasnjeni su detaljno svi koraci algoritma i kako su oni realizovani. Objasnjeni su problemi detekcije značajki na slici, praćenje tih značajki kroz sekvencu slika i na kraju kako je estimirana pozicija kamere. Pozicija kamere je estimirana prvo pronalaženjem esencijane matrice, zatim odstranjivanjem vanpopulacijskih značajki i na kraju procjena ukupne matrice rotacije i translacije u svakom okviru. U sljedećem poglavlju je opisano koji je to programski jezik korišten za implementaciju algoritma, koje značajne biblioteke za kompjutersku viziju su korištene, te su predstavljeni i testni podaci pomoću kojih je evaluiran algoritam monokularne vizualne odometrije. Na kraju su prikazani i eksperimentalni rezultati.

## Poglavlje 4.

# Programska implementacija

U prethodnom poglavlju je opisan algoritam monokularne vizualne odometrije detaljno, poglavlje 3. U ovom poglavlju će biti objašnjeno koji je programski jezik korišten, koje biblioteke za kompjutersku viziju. Također, bit će predstavljeni i podaci pomaloću kojih će algoritam biti testiran. Na kraju, u zadnjem potpoglavlju će biti prikazani i eksperimentalni rezultati.

### 4.1. Softversko rješenje

Kompjuterska vizija je disciplina u kojoj se omogućava računaru da replicira ljudski vid. To je jedna od grana vještacke inteligencije koja sakuplja informacije iz slika ili video sekvence tako što ih procesuira i pronađe odgovarajuće atributе. Cijeli proces uključuje prikupljanje slika, prikazivanje, analiziranje, identificiranje i izvlačenje informacija. Projekat kompjuterske vizije vrši prevodenje digitalnog vizualnog sadržaja u multidimenzionalne podatke. Ovi podaci su dalje prevedeni u jezik koji je razumljiv računaru. Glavni cilj mašine je da sakuplja infomracije iz piksela. Potrebno je izabrati odgovarajući programski jezik koji može procesuirati komplikovane algoritme. Postoji preko 1500 programskih jezika u svijetu koji se koriste, a programski jezici koji se najviše koriste za realiziranje algoritama kompjuterske vizije su: BASIC/Pascal, LISP, Hardware Description Languages (HDL-s), Java, MATAB, Python, C/C++ itd.

U ovom radu je korišten **C++ programski jezik**<sup>1</sup>. C++ je jedan od najkorištenijih programskih jezika na svijetu. C porodica jezika se jako mnogo koristi za realiziranje algoritama vještacke inteligencije, samam tim i za kompjutersku viziju. Mnoge hardvarske biblioteke koriste upravo ove jezike. Osnovna open source biblioteka kompjuterske vizije, OpenCV, je napisana u ovim jezicima, mada OpenCV ima podršku i za druge programske jezike, kao što je MATLAB, Java i Python, tako da se ovi jezici pored jezika C porodice najviše koriste u kompjuterskoj viziji. Sama OpenCV biblioteka bit će detaljno opisana u nastavku dokumenta.

Prednosti C++ programskog jezika za korištenje u kompjuterskoj viziji su:

- Besplatan je! Veliki dio OpenCV biblioteke je open source.
- Sadrži veliku kolekciju AI biblioteka i alata.

---

<sup>1</sup><https://isocpp.org/>

- Velika brzina izvršenja.
- Objektno-orjentirani principi koji su korisni za organiziranje podataka.
- Podrška za mnoge platforme i uređaje.
- Veliki broj različitih korisnika (oko 47000), istraživača, te ljudi iz različitih područja industrije.

Nedostaci C++ programskog jezika za korištenje u kompjuterskoj viziji:

- Težak je za početnike koji nemaju nikakvog iskustva u korištenju C++ ili OpenCV.
- OpenCV dokumentacija nije toliko sadržajna.
- Vizualizacija i debagiranje je teško u C++ okruženju.

#### 4.1.1. OpenCV

**OpenCV** (eng. *Open Source Computer Vision*)<sup>2</sup> skup biblioteka otvorenog koda koji je pokrenut od strane američke tvrtke Intel (eng. *Integrated Electronics Corporation*) 1999. godine u svom istraživačkom centru u Rusiji. Biblioteke su višeplatformske (eng. *cross-platform, multiplatform*), mogu se koristiti u raznim aplikacijama, na raznim operativnim sistemima (Windows, Linux, Mac OS X,..). OpenCV služi raznim korisnicima da iskoriste i preoblikuju vlastiti kod. Biblioteka sadrži više od 2500 već gotovih optimiziranih algoritama, kao i modul Machine learning, fokusiran na prepoznavanje uzoraka (eng. *pattern recognition*) i klasteriranje. Algoritme možemo koristiti za pronalaženje, odnosno otkrivanje i prepoznavanje lica. Također se koristiti i za prepoznavanje objekata, praćenje objekata u pokretu itd. [37].

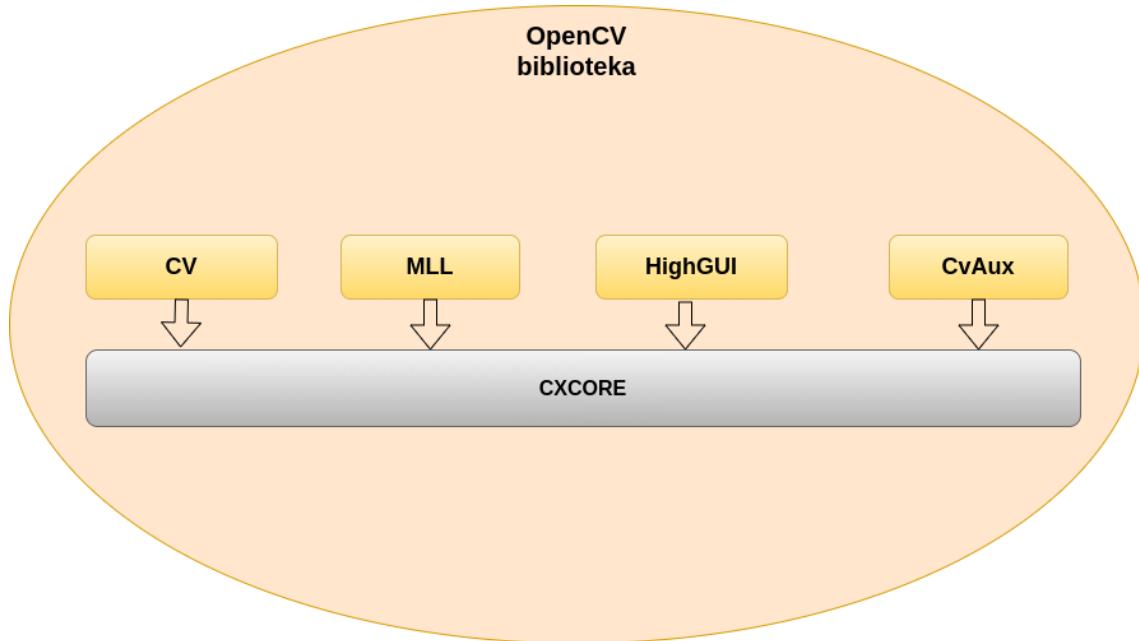
Osnovna zadaća OpenCV-a pružanje je jednostavne platforme za rad sa računarskim vidom i omogućivanje aplikacija koje koriste računarski vid. Zbog komplikovanih algoritama koji rade sa velikim brojem operacija i podataka zahtjeva se optimiziran kod napisan na nižem nivou zbog veće brzine izvršavanja. Zbog toga su algoritmi OpenCV biblioteka originalno pisani u C/C++ programskom jeziku. Osim ranije navedenih jezika, postoje sučelja prema većini popularnih jezika, kao što su Java, Python i MATLAB. Bibliotku je moguće koristiti i za izradu mobilnih Android i iOS aplikacija. Zajednica OpenCV korisnika okuplja 50 hiljada ljudi koji su napravili oko 14 miliona preuzimanja. Najčešći korisnici su tvrtke, istraživačke skupine, državna tijela, studenti itd. OpenCV licenca je strukturirana tako da je moguće razvijati komercijalne aplikacije koristeći cijelu biblioteku ili neki njen dio [37].

OpenCV u nekim od svojih proizvoda koriste mnoge od najvećih IT tvrtki - Microsoft, Google, Intel, IBM, Honda, Sony te istraživački centri - Stanford, MIT, Cambridge itd. Konkretni primjer primjene je spajanje slika u Streetviewu (eng. *image stitching*), kontrola robota (navigacija u prostoru, interakcija s objektima), provjera deklaracija proizvoda u industriji, praćenje objekata sigurnosnim kamerama [37].

---

<sup>2</sup><https://opencv.org/>

Biblioteka OpenCV podijeljena je u pet osnovnih komponenti, što je prikazano na slici 4.1.



Slika 4.1: Struktura OpenCV-a

**CXCORE:** Sadrži osnovne strukture podataka i algoritme, podršku za jezik XML i crtanje. To uključuje operacije na kolekcijama, dinamičke strukture podataka (red, skup, graf, stablo), pohranjivanje podataka (eng. *data persistance*), obradu gresaka i sistemske funkcije.

**CV:** Najopsežniji dio biblioteke. Sadrži algoritme za obradu slike. Također, ovdje se nalaze algoritmi kompjuterskogvida višeg nivoa apstrakcije. Upravo ovaj dio biblioteke najviše korišten za implementaciju VO algoritma.

**MLL:** (eng. *Machine Learning Library* - MLL) je skup klase i funkcija koje možemo koristiti za klasifikaciju, regresiju i klasteriranje podataka.

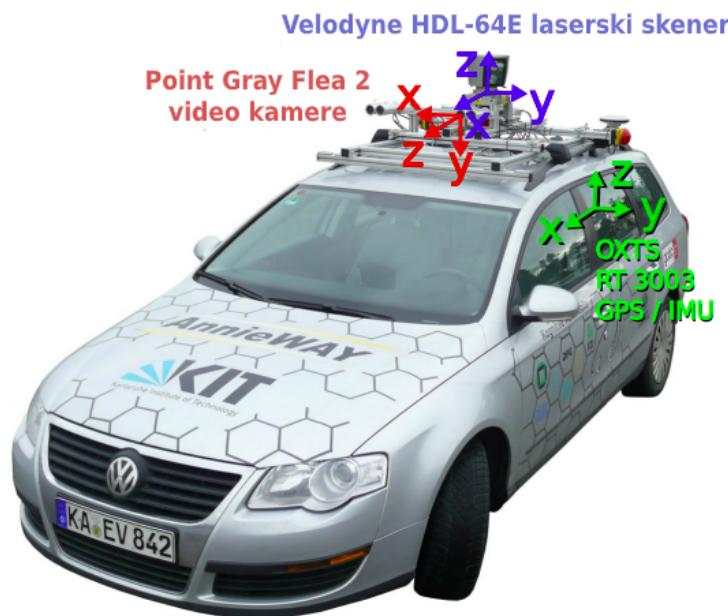
**HighGUI:** Iako je OpenCV u većini slučjeva korištena u aplikacijama bez eksplicitnog korisničkog sučelja ili aplikacija čiji je UI razvijen u nekom od razvojnih okvira (npr. WinForms, Qt, JavaFX), ponekad je korisno imati mehanizam pomoću kojeg je lako brzo vizualizirati rezultate primjenjenih algoritama. HighGUI se koristi upravo u tu svrhu. On pruža jednostavno sučelje za kreiranje prozora, crtanje, procesiranje jednostavnih korisničkih akcija i manipulaciju slikama.

**CvAux:** Ova komponenta sadrži algoritme koji više nisu u aktivnom dijelu biblioteke (npr. Hidden Markov Model prepoznavanje lica), te eksperimentalne algoritme (npr. prepoznavanje pozadinskog i prednjeg segmenta slike).

#### 4.1.2. KITTI podaci

Za evaluaciju algoritma korišteni su **KITTI podaci**. U ovom potpoglavlju će biti opisano kakvi su ti podaci i od čega se sastoje. Podaci su snimljeni za potrebe evaluiranja i testiranja algoritama kompjuterske vizije. Podaci su open source i mogu se pronaći na oficijalnoj stranici "The KITTI Vision Benchmark Suite"<sup>3</sup>. Ovi podaci su projekat Karlsruhe Institute of Technology i Toyota Techological Institute at Chicago.

Podaci su snimljeni sa VW Passat auta (slika 4.2) i koriste se za istraživanja u mobilnoj robotici i autonomnoj vožnji, kao što je već navedeno. Snimani su različiti scenariji u saobraćaju šest sati brzinom od 10-100 Hz, te su korišteni različiti senzori, stereo kamere u boji i grayscale visoke rezolucije, Velodyne 3D laserski skener, te GPS/IMU inercijski navigacijski sistem visoke preciznosti. Scenariji su različiti, snimljene su stvarne scene u saobraćaju, otvoreni putevi kroz ruralna područja i scene iz gradskog prometa sa mnogo statičkih i dinamičkih objekata u Karlshure, Njemačka. Podaci su kalibrirani, sinhronizovani sa preciznim vremenima, te su priložene slike sa kojih je uklonjena distorzija i slike sa kojih nije. Glavna svrha ovih podataka je da omoguće dalja istraživanja u kompjuterskoj viziji i istraživanje algoritama za autonomnu vožnju [12].



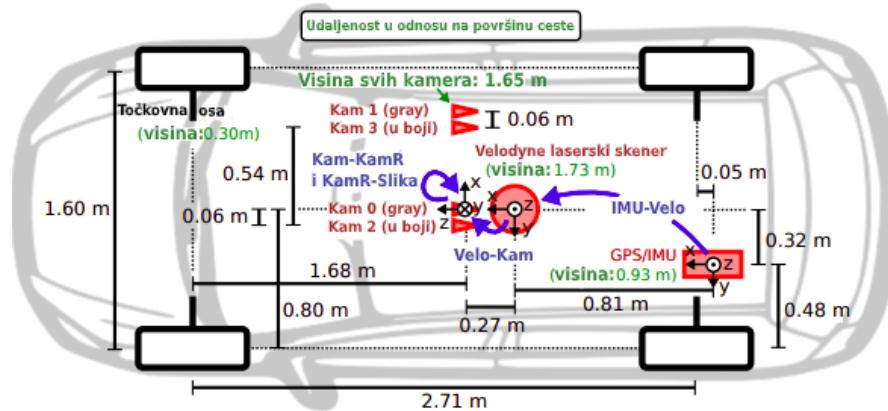
Slika 4.2: Platforma za snimanje podataka: VW Passat sa senzorima [12]

Raspored senzora na VW Passat autu je sljedeći, slika 4.3, [12]:

- Dvije PointGray Flea2 grayscale kamere (FL2-14S3M-C) sa karakteristikama: 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter.
- Dvije PointGray Flea2 kamere u boji (FL2-14S3C-C) sa karakteristikama: 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter.
- Četiri Edmund Optics lenses sa karakteristikama: 4mm, opening angle  $\sim 90^\circ$ , vertical opening angle of region of interest (ROI)  $\sim 35^\circ$ .

<sup>3</sup><http://www.cvlibs.net/datasets/kitti/index.php>

- Jedan Velodyne HDL-64E rotirajući laserski skener sa karakteristikama: 10 Hz, 64 beams, 0.09 degree angular resolution, 2 cm distance accuracy, collecting  $\sim 1.3$  million points/second, field of view: 360° horizontal, 26.8° vertical, range: 120 m.
- Jedan OXTS RT3003 inercijalni i GPS navigacijski sistem sa karakteristikama: 6 axis, 100 Hz, L1/L2 RTK, resolution: 0.02m / 0.1°.
- Dva PC računara sa dva six-core Intel XEON X5650 procesora, RAID 5 hard disk sa kapacitetom od 4 terabajta. Na računaru je Linux operativni sistem (64 bit) i real-time baza podataka za pohranu podataka.



**Slika 4.3:** Raspored senzora na autu VW Passat: Dimnizije i mjesta gdje su senzori postavljeni u odnosu na ram vozila (crveno), te udaljenost od zemlje je označena zelenom bojom. Transformacije između senzora su prikazane plavom bojom [12]

Snimljeni su različiti podaci sa različitih senzora, kao što je već navedeno, "sirovi" podaci, te obrađeni podaci (kalibrirane slike itd.). Za evaluaciju konkretno vizualne odometrije i SLAM-a priložene su pogodne sekvene slika<sup>4</sup>.

Odometry benchmark se sastoји од 22 stereo sekvene slike, spremljениh u .png formatu. Prvih jedanest sekvenci (00-10) imaju i stvarnu trajektoriju (eng. *ground truth*). Ovim podacima se može testirati dakle monokularna ili stereo odometrija. Ukoliko se testira monokularna odometrija onda se koriste slike sa jedne kamere, a ne sa obje. Dakle moguće je skinuti: stereo sekvene u boji ili grayscale, kalibracijske podatke, podatke o stvarnoj trajektoriji vozila i laserske velodyne podatke.

Za testiranje konkretno ovog algoritma korištene su slike sekvene 00 sa jedne od grayscale kamere koje su već kalibrirane. Sekvenca 00 sadrži 4540 slika. Podaci o stvarnoj trajektoriji su korišteni za računanje skalirajućeg faktora koji se koristi pri računanju ukupne matrice translacije na način kako je to prikazano u jednačini 3.10. Također, ground truth se koristi kasnije i za računanje greške koja se javlja pri estimacije trajektorije ovim algoritmom monokularne vizualne odometrije. Ground truth trajektorija je sadržana u 00.txt fajlu. To je zapravo matrica sa 12 kolona i 4540 redova. U svakom redu matrice je

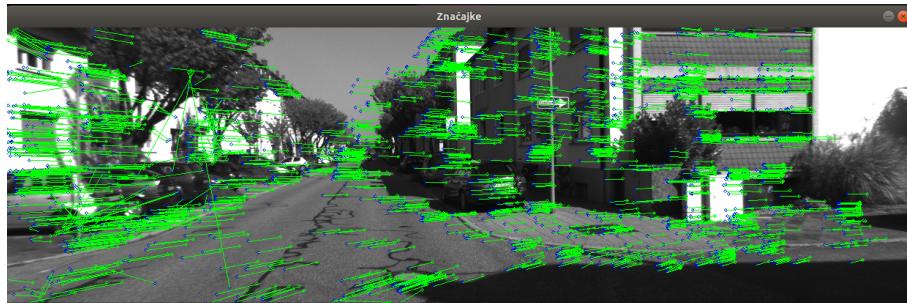
<sup>4</sup>[http://www.cvlibs.net/datasets/kitti/eval\\_odometry.php](http://www.cvlibs.net/datasets/kitti/eval_odometry.php)

smještena  $3 \times 4$  matrica stvarnih transformacija koordinata između frejmova (umnožak matrice rotacija i matrice translacije između frejmova). Prve četiri vrijednosti u redu matrice predstavljaju prvi red  $3 \times 4$  matrice stvarnih transformacija koordinata, i dalje slično do dvanaestog člana reda. Za računanje greške estimirane trajektorije VO algoritma izvršene su određene transformacije ovih ground truth podataka. Izvučene su informacije o stvarnom vektorima translacije, kao i informacije o matricama rotacija koja su pretvorene u Eulerove uglove ZYX (yaw, pitch, roll).

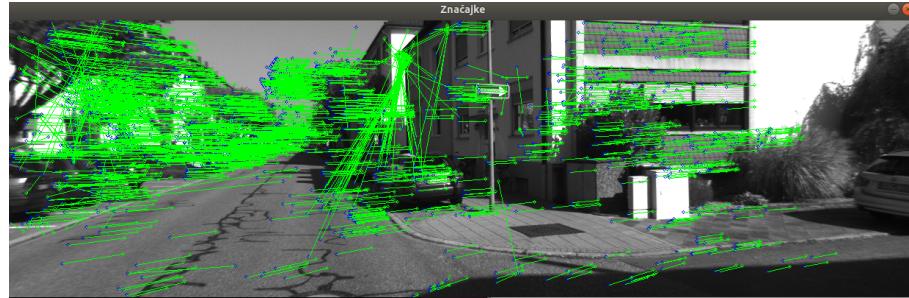
## 4.2. Implementacija algoritma

U ovom dijelu je predstavljen algoritam monokularne vizualne odometrije za lokalizaciju i njegovi koraci. Slike koje se koriste su već kalibrirane tako da nije potrebno vršiti kalibraciju slika. Dalje ove slike se učitavaju jedna za drugom u kod gdje se obavlja i lokalizacija simultano. Razlog što se koriste već kalibrirane slike je upravo taj što algoritam onda ne mora gubiti vrijeme i na kalibriranje. U trećem poglavljju ovog rada je spomenuto da će biti korištena tri različita detektora linijskih značajki: Line Segment Detector, Standard Hough Line Transform, Probabilistic Hough Line Transform. Naime, algoritam je pokrenut sa svakim detektorem pojedinačno. Cilj je uporediti sva tri rezultatata da bismo otkrili koji od detektora ima najbolje performanse za jedan ovakav izvor podataka. Koraci ovog algoritma za lokalizaciju su:

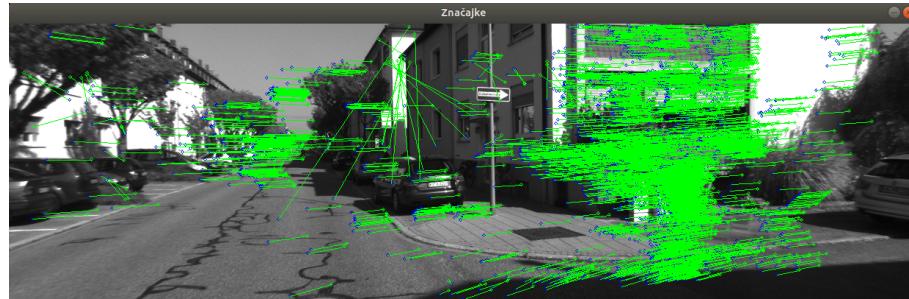
1. Učitaj kalibriranu grayscale sliku jedne od kamere 00 KITTI sekvence.
2. Detektuj linijske značajke i njihove krajnje tačke LSD ili PHLT ili SHLT detektorm.
3. Prati ekstraktovani set značajki na sljedećoj slici koristeći KLT tracker.
4. Izračunaj esencijalnu matricu i odbaci moguće outliere koristeći RANSAC.
5. Izračunaj trenutnu matricu rotacije i translacije iz estimirane esencijalne matrice.
6. Ažuriraj ukupnu poziciju nadovezivanjem trenutne matrice rotacije i translacije, računajući i skalirajući faktor za translaciju.
7. Nacrtaj detektovane značajke na slike i prikaži njihove kretanje.
8. Spremi informaciju o ukupnoj poziciji u .txt fajl i počni ponovo za narednu sliku isti proces.



Slika 4.4: Značajke i njihovo pomjeranje: LSD detektor

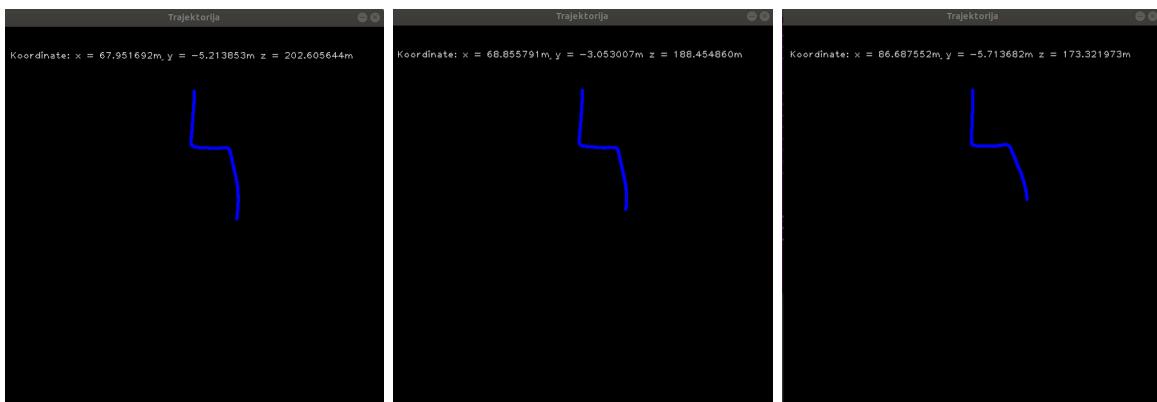


Slika 4.5: Značajke i njihovo pomjeranje: PHLT detektor



Slika 4.6: Značajke i njihovo pomjeranje: SHLT detektor

Na slikama su prikazane značajke i njihovo pomjeranje za sva tri detektora, slike 4.4, 4.5 i 4.6 respektivno. Ovo ilustrira kretanje kamere. Moguće je uočiti pojedine outliere na svakoj slici, ali većina ovih outliera će biti otklonjena RANSAC metodom pri procjeni esencijalne matrice. Očigledno je i da svaki detektor različito detektuje značajke. Vidljivo je da LSD detektor ravnomjernije detektuje značajke po cijeloj slici u odnosu na PHLT i SHLT detektore. Simultano se na dodatnom prozoru iscrtava i trajektorija kretanja kamere u x-z ravni što je prikazano na slici 4.7. Vidimo da su trajektorije sličnog oblika već za nekoliko prvih slika, ali vidimo da postoji i razlika među njima. U narednom poglavlju će biti provjereno koja trajektorija ima bolju tačnost, računajući grašku između ovih estimiranih trajektorija i ground truth trajektorije.



Slika 4.7: Iscrtavanje trajektorije kretanja kamere na prozoru u x-z ravni. S lijeva na desno: LSD detektor, PHLT detektor i SHLT detektor

### 4.3. Eksperimentalni rezultati

Da bi se provjerila tačnost ovog implementiranog algoritma za lokalizaciju provedeni su određeni eksperimenti. U potpoglavlju će biti predstavljeni ti eksperimenti i njihovi rezultati. Prikupljeni podaci o trajektoriji kretanja kamere, tj. pozicija i orijentacija su poređeni sa stvarnom trajektorijom, ground truth. Podaci su smješteni u txt fajlove. Obrađeni su u programskom jeziku i okruženju MATLAB, jer je riječ o velikim maticama, a s njima je najlakše manipulirat u MATLAB-u. Najprije su podaci o poziciji i orijentaciji iz oba seta podataka, estimirana trajektorija VO algoritma i ground truth prilagođeni. Izvučene su informacije o poziciji ( $x$ ,  $y$  i  $z$  koordinate), te informacije o orijentaciji (Eulerovi uglovi ZYX: yaw, pitch, roll). Nadalje, ti podaci su upoređeni. Kvalitet VO algoritma računajući ATE (eng. *Absolute Trajectory Error*) i relativnu grešku između estimirane trajektorije i ground truth trajektorije. Može se se računati srednja ATE greška translacije kao što je prikazano u jednačini 4.1, gdje je  $N$  broj pozicija,  $\Delta t_x$ ,  $\Delta t_y$  i  $\Delta t_z$  absolutna razlika između ground truth translacije i translacije estimirane putanje.

$$ATE_{trans} = \frac{1}{N} \sum_{i=0}^{N-1} (|\Delta t_{xi}| + |\Delta t_{yi}| + |\Delta t_{zi}|) \quad (4.1)$$

Slično se računa i za ATE grešku rotacije, jednačina 4.2, gdje je  $N$  broj pozicija,  $\Delta\phi$  (roll),  $\Delta\theta$  (pitch) i  $\Delta\psi$  (yaw) absolutna razlika između ground truth rotacije i rotacije estimirane putanje.

$$ATE_{rot} = \frac{1}{N} \sum_{i=0}^{N-1} (|\Delta\phi_i| + |\Delta\theta_i| + |\Delta\psi_i|) \quad (4.2)$$

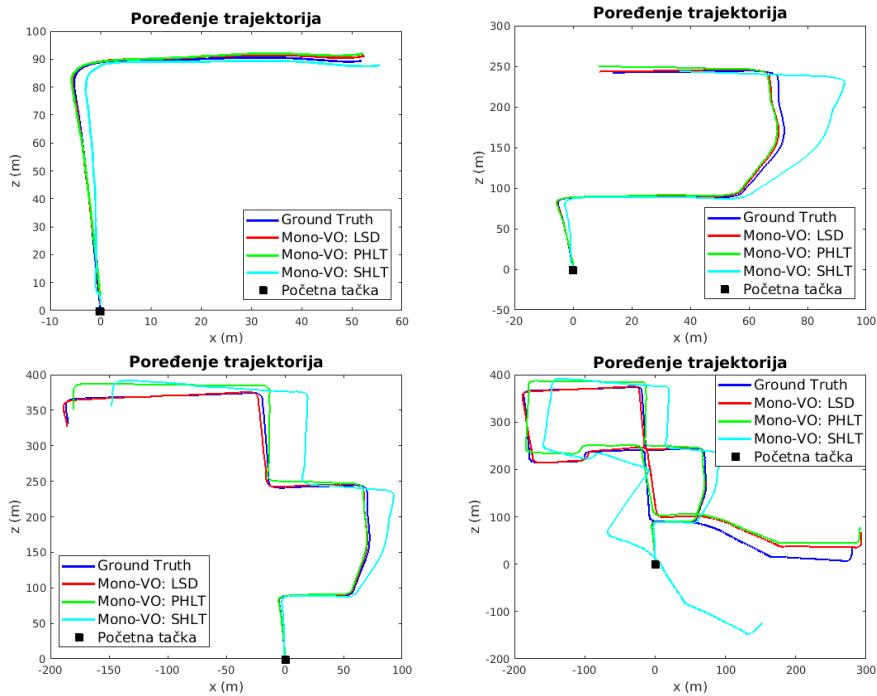
Relativna greška za translaciju i rotaciju između ground truth trajektorije i estimirane trajektorije se računa kako je prikazano u jednačinama 4.3 i 4.4, gdje je  $|t_{gx}|$ ,  $|t_{gy}|$  i  $|t_{gz}|$  translacija ground truth trajektorije.

$$\epsilon_{trans} = \frac{|\Delta t_x| + |\Delta t_y| + |\Delta t_z|}{|t_{gx}| + |t_{gy}| + |t_{gz}|} \quad (4.3)$$

$$\epsilon_{rot} = \frac{|\Delta\phi| + |\Delta\theta| + |\Delta\psi|}{|t_{gx}| + |t_{gy}| + |t_{gz}|} \quad (4.4)$$

Ove greške su izračunate za više različitih pređenih distanci, tj. broj frejmova. Tako će u nastavku biti prikazane trajektorije koje su dobivene za svaki detektor za različit broj frejmova (200, 500, 1000 i 2000 frejmova respektivno), slika 4.8. Sa ovih grafika već možemo vidjeti kakva je tačnost pojedinih algoritama sa određenim detektorom.

Na slici 4.9 će biti prikazana trajektorija za sve frejmove (4540 frejmova), tj. cijela trajektorija. Na ovoj slici također se može uočiti ponašanje svakog algoritma sa određenim

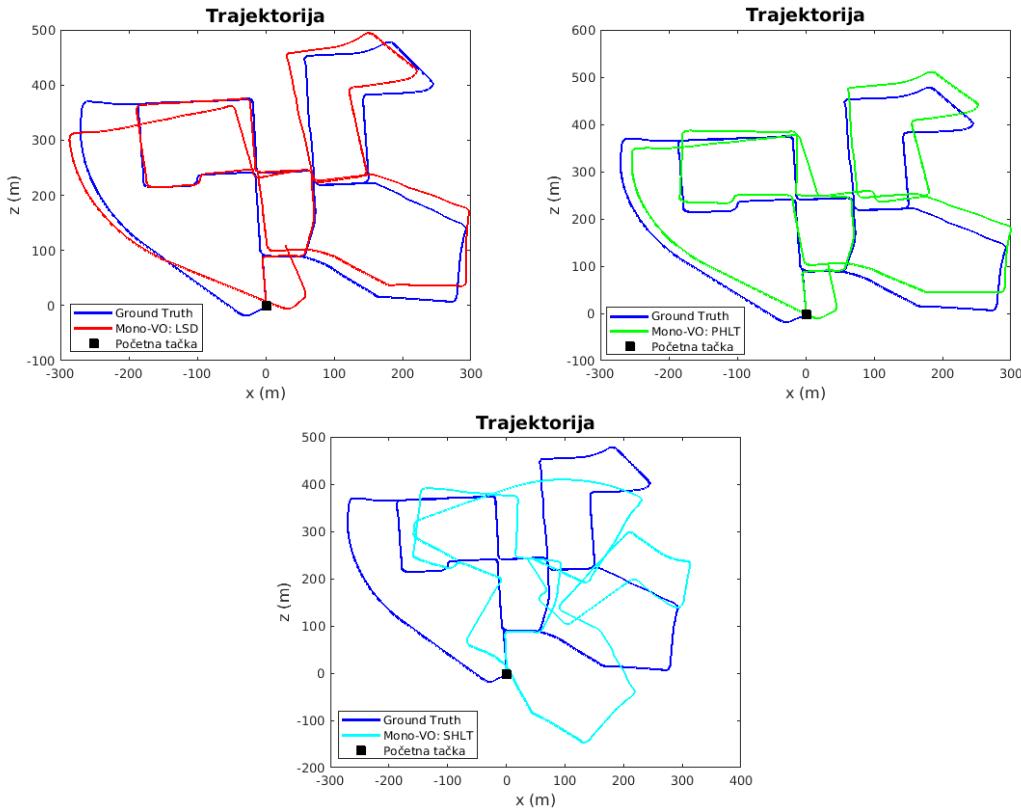


Slika 4.8: Trajektorije u x-z ravni za različit broj frejmova 200, 500, 1000 i 2000 respektivno

detektorom, već je očigledno da algoritam sa SHLT detektorom daje jako loše rezultate, dok su trajektorije sa LSD i PHLT detektorom slične i daju bolje rezultate.

Nadalje će biti prikazani grafici proračunatih relativnih greški za rotaciju, slika 4.10, i translaciju, slika 4.11, za različit broj frejmova. Za 200 frejmova je vozilo prešlo otprilike 100 metara od ukupne predene udaljenosti koja iznosi 800 metara. Računate su greške i za veći broj frejmova, za veću pređenu distancu. Bitno je pomenuti da se greška povećava vremenom i kako vozilo prelazi veću udaljenost. Međutim, da bi bilo provjereno koji algoritam, s kojim detektorom daje bolje rezultate dovoljno je da se izračunaju srednje greške za 200 frejmova. Greške i njihove vrijednosti će biti prikazane u tabeli u nastavku. Na graficima koji su prikazani na slikama 4.10 i 4.11 se vidi da je rotacijska greška ima veće vrijednosti nego translacijska greška. LSD detektor ima puno manje vrijednosti greška za sve provjerene udaljenosti, dok PHLT i SHLT detektor imaju velike vrijednosti rotacijske greške. PHLT detektor ima iznenadni skok na početku sekvene kod rotacijske greške. Za veći broj frejmova greška se povećava za sve detektore, kako translacijska, tako i rotacijska greška, što se može vidjeti na grafiku gdje se greška računa za 2000 frejmova. Naime, iz grafika se jasno vidi da se LSD detektor ponaša mnogo bolje nego PHLT i SHLT, čak PHLT i SHLT imaju jako loše rezultate. Na 2000 frejmova, na pola cijelokupne putanje 400 m, vidi se da svi detektori imaju velike translacijske i rotacijske greške. Na slikama koje su prikazivale raspored značajki na trenutnoj slici, slike 4.4, 4.5 i 4.6, uočeno je da LSD detektor itekako ravnomjernije detektuje značajke na slici i sa puno manje outliera, pa ovi rezultati upravo to pokazuju. U narednoj tabeli 4.1 će biti prikazane srednje vrijednosti ATE i relativne greške za svaki detektor. Tako je moguće napraviti lakšu usporedbu rezultata za svaki detektor.

Iz izračunatih grešaka iz tabele 4.1 može se uočiti da LSD detektor ima mnogo bolje rezultate nego detektori PHLT i SHLT koji imaju velike greške naročito rotacijske greške.



Slika 4.9: Ukupna trajektorija u x-z ravni

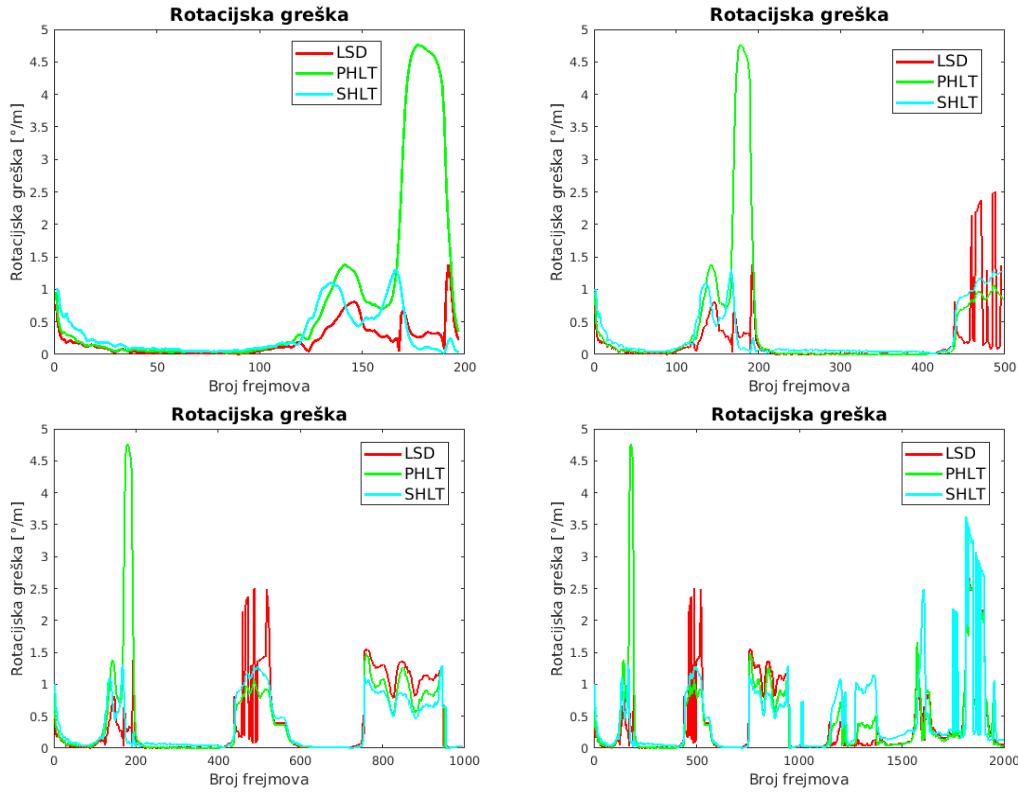
Može se uočiti da PHLT detektor ima manju translacijsku grešku, ATE i relativnu grešku, ali ima veću rotacijsku grešku od SHLT detektora. Jasno je da oba detektora PHLT i SHLT daju jako loše rezultate i da LSD detektor mnogo bolje detektuje linijske značajke ravnomjerno po svim slikama sa puno manje outliera. Logično je da se za ovakve aplikacije, na osnovu svega navedenog, koristi LSD detektor.

Tablica 4.1: ATE i relativne greške za 200 frejmova

Detektor	$ATE_{trans}$ [m]	$ATE_{rot}$ [ $^{\circ}$ ]	$\epsilon_{trans}$ [%]	$\epsilon_{rot}$ [ $^{\circ}/m$ ]
LSD	1.06	4.27	0.60	1.67
PHLT	3.55	30.37	0.65	6.48
SHLT	5.20	21.63	0.78	2.50

Na slikama 4.12 i 4.13 prikazana je greška po pojedinim koordinatama i uglovima koristeći boxplot za 200 frejmova. Prikazana je absolutna razlika između koordinata i uglova ground truth trajektorije i VO trajektorije. Ovi boxplot grafici su prikazani da bi se moglo lakše uočiti gdje se javlja greška i zašto je ona za pojedine detektore velika.

Na slici 4.12 može se vidjeti da je absolutna greška za pojedine ose za detektore SHLT i PHLT uglavnom veća u odnosu na LSD detektor. Može se očitati u kojem rangu se

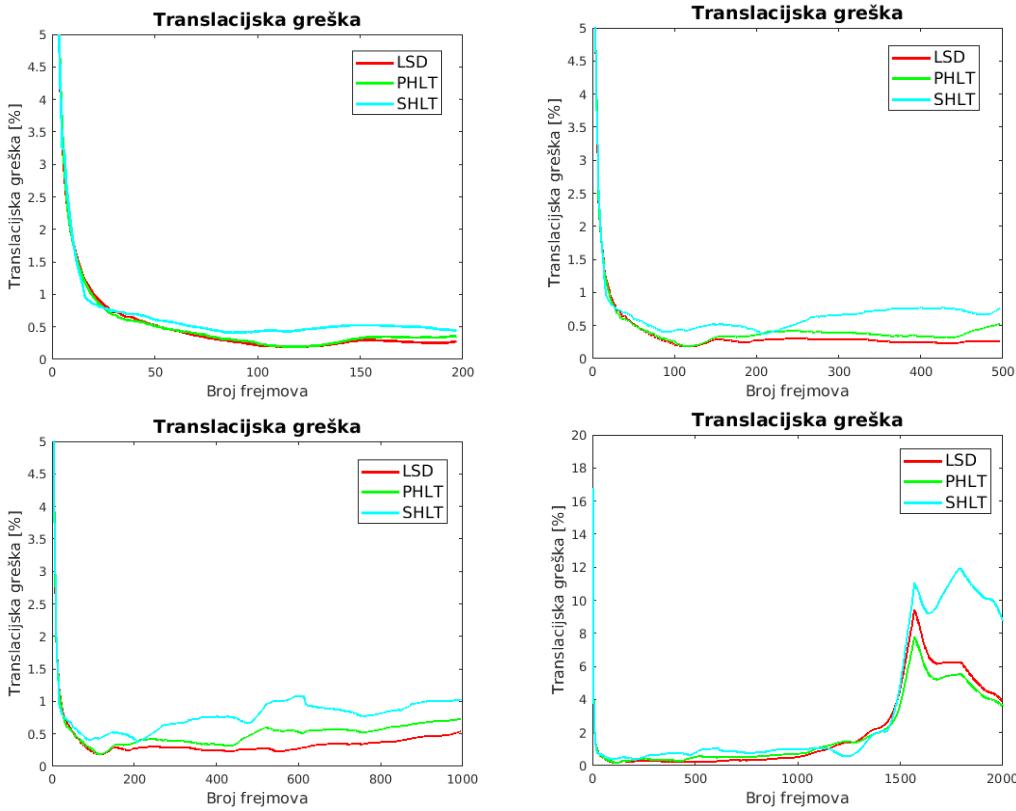


**Slika 4.10:** Relativna rotacijska greška izražena u  $[\circ/m]$  u odnosu na različit broj frejmova

greška kreće i koja je njena median vrijednost za svaku osu pojedinačno x, y i z.

Na slici 4.13 može se vidjeti da se apsolutne greške za pojedine uglove kreću u velikom rangu sa puno outlier vrijednosti. Ponovo se može uočiti da LSD detektor ima mnogo manju grešku i po pojedinim uglavima sa puno manje outlier vrijednosti u odnosu na PHLT i SHLT detektor. PHLT detektor ima veću grešku po pojedinim uglovima u odnosu na SHLT detektor. Mnogo je outlier vrijednosti kod rotacijske greške za pojedine uglove, roll, pitch i yaw, te se greške kreću u velikom rangu sa velikim median vrijednostima za detektore PHLT i SHLT, pa iz toga proizilaze i velike ATE i relativne greške. Na ovim boxplot graficima se može uočiti da je za sve detektore greška po roll i yaw uglovima mnogo veća nego po pitch uglu.

Nakon svih proračunatih grešaka i analiza jasno je da LSD detektor ima najbolji rezultat. LSD detektor obavlja nabolju detekciju linijskih značajki ravnomjerno po cijeloj slici i za ovaj set slika je dao najbolje rezultate, kada se vozilo kreće ulicom u nekom urbanom području. Ovaj algoritam monokularne vizualne odometrije sa LSD detektorom iscrtava trajektoriju i estimira poziciju kamere, robota ili vozila sa određenom greškom. Greška se s povećanjem distance koju vozilo prelazi i stvara se određeni drift između stvarne trajektorije vozila i estimirane trajektorije, ali ipak je greška LSD detektora u razumnom opsegu. Postoje neki algoritmi kojima je moguće ukloniti drift i dobiti mnogo bolju tačnost algoritma ali oni nisu implementirani u ovom radu. Ovi algoritmi obrađuju jako puno podataka i video sekvene sa velikom brzinom pa se obično implementiraju na GPU sa višejezgrenim najnovijim procesorima ukoliko se koriste u industriji, kao npr. za autonomna vozila. Ovaj algoritam monokularne vizualne odometrije je implementiran na CPU sa Intel i5 jednojezgrenim procesorom na Linux operativnom sistemu.

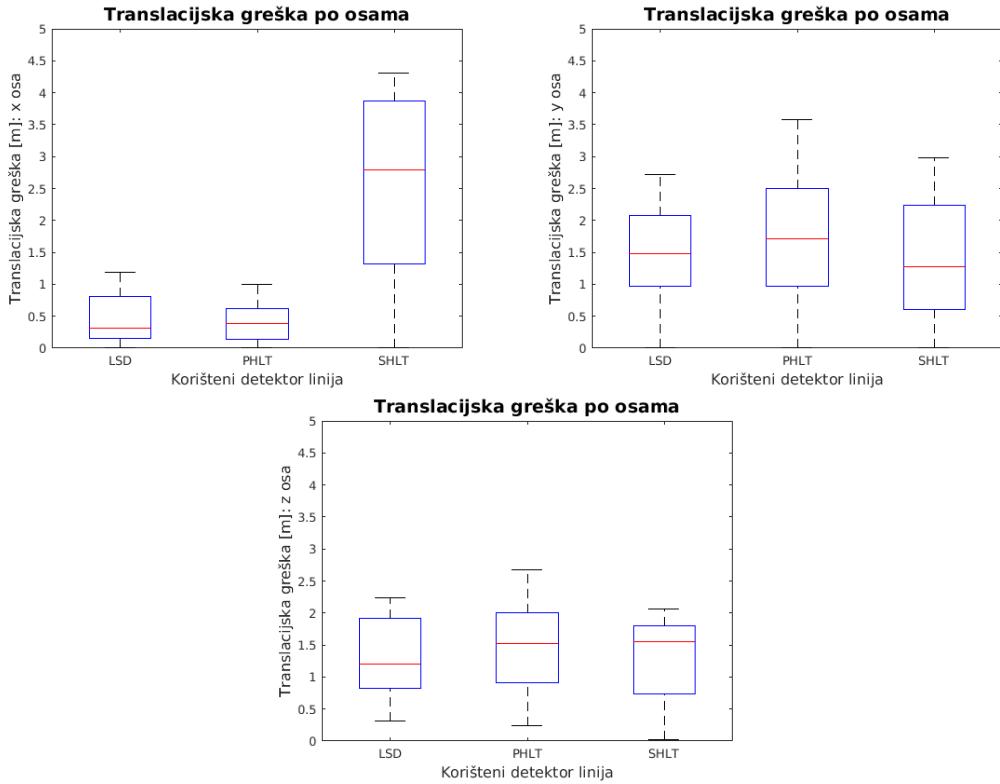


**Slika 4.11:** Relativna translacijska greška izražena u [%] u odnosu na različit broj frejmova

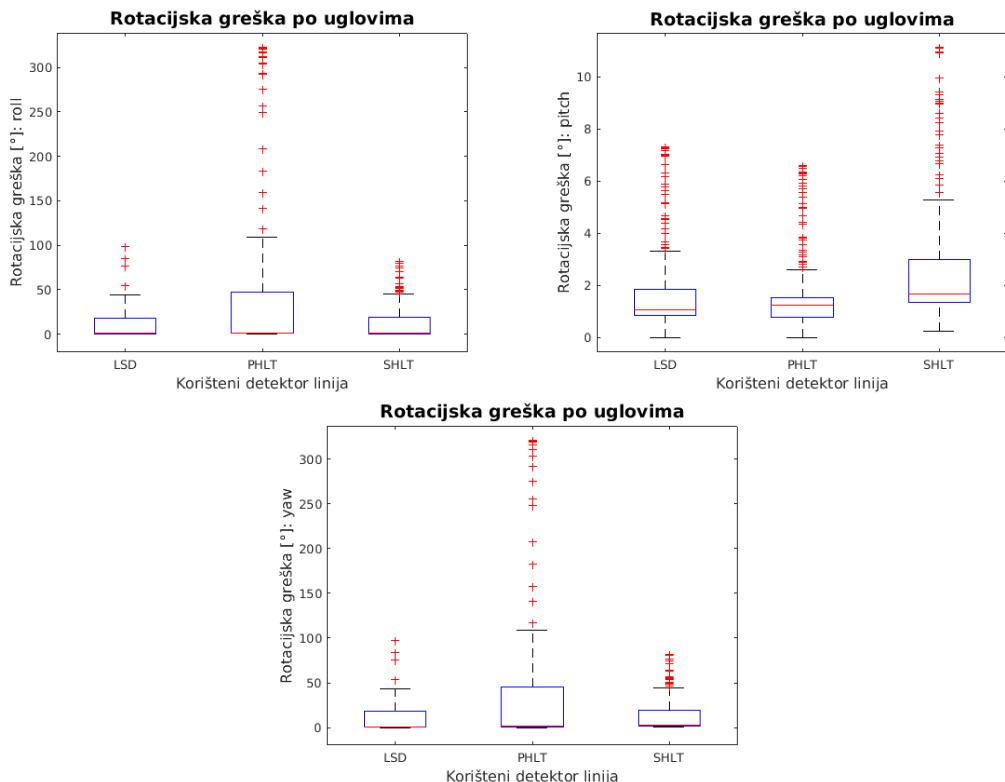
Na kraju je provjereno i vrijeme izvršavanja algoritma sa odgovarajućim detektorom. Ova vremena izvršavanja su prikazana u tabeli 4.2. Algoritam monokularne vizualne odometrije sa LSD detektorom ima malo sporije vrijeme izvršenja od algoritma monokularne vizualne odometrije sa PHLT detektorom koji ima najmanje vrijeme izvršavanja. Također, algoritam monokularne vizualne odometrije sa PHLT detektorom ima najsporije vrijeme izvršenja, pa vidimo da i po ovom parametru ima najlošiji rezultat. Prikazana vremena izvršenja su vremena izvršenja algoritma nad cijelom sekvencom slika (4540 slika).

**Tablica 4.2:** Vremena izvršavanja algoritama sa LSD, PHLT i SHLT detektorma

Vrijeme izvršavanja	Mono VO: LSD	Mono VO: PHLT	Mono VO: SHLT
Ukupno vrijeme izvršavanja	2679.28 sekundi	2023.65 sekundi	3906.06 sekundi
Prosječno vrijeme izvršavanja	0.59 sekundi	0.44 sekundi	0.86 sekundi



Slika 4.12: Translacijska greška izražena u [m] po pojedinim osama za sve detektore



Slika 4.13: Rotacijska greška izražena u [°] po pojedinim uglovima za sve detektore

## 4.4. Zaključak

U ovom poglavlju je opisano koji programski jezik korišten za implementaciju algoritma, te biblioteka kompjuterske vizije OpenCV i njene karakteristike. Nakon toga su opisani i testni podaci kojima je evaluiran algoritam. Opisan je i finalni algoritam po koracima, te su na kraju priloženi eksperimentalni rezultati. Zaključak koji može biti donesen na osnovu eksperimentalnih rezultata je da LSD detektor za određene predene udaljenosti ima najbolje rezultate od sva tri korištena detektora. Bitno je spomenuti da sva tri detektora kad vozilo pređe veliku udaljenost prave veliku grešku pri estimaciji trajektorije. Ovaj drift između stvarne trajektorije i estimirane trajektorije ovaj algoritam ne može sam ukloniti bez implementacija dodatnih algoritama.

Algoritam vizualne odometrije je implementiran već sa detektorima tačaka kao značajki (eng. *point features*). Jedan od najviše korištenih algoritama monokularne vizualne odometrije je Libviso Mono od Andreasa Geigera<sup>5</sup>. Ovaj algoritam je baziran na značajkama (eng. *feature-based*) kao i algoritam koji je realizovan u ovom završnom radu. Libviso Mono je predstavljen i na KITTI Vision Benchmark Suite stranici, te njegove performanse [38].

Relativna rotacijska greška Libviso Mono algoritma za cijelu KITTI sekvencu 00 (4540 frejmova) iznosi  $0.0209 \text{ } \circ/m$ . Relativna translacijska greška iznosi (za 4540 frejmova) 11.91 %. Relativne rotacijske greške implementiranog Mono VO algoritma u ovom radu (za 4540 frejmova) za LSD, PHLT i SHLT detektor iznose:  $3.33 \text{ } \circ/m$ ,  $3.38 \text{ } \circ/m$  i  $9.52 \text{ } \circ/m$ , respektivno. Relativne translacijske greške implementiranog Mono VO algoritma za LSD, PHLT i SHLT detektore (za 4540 frejmova) iznose: 9.36 %, 9.94 % i 18.42 %, respektivno. Ukoliko se uporedne oba algoritma, Libviso Mono i implementirani Mono VO algoritam jasno je da Libviso Mono ima manju rotacijsku grešku, dok Mono VO algoritam sa LSD i PHLT detektorima ima manju translacijsku grešku. Mono VO algoritam sa SHLT detektorom ima mnogo veću vrijednost translacijske i rotacijske greške od Libviso Mono algoritma. Prosječno vrijeme izvršavanja Libviso Mono algoritma iznosi 0.268 sekundi, dok prosječno vrijeme izvršavanja Mono VO algoritma iznosi 0.59 sekundi, 0.44 sekundi i 0.86 sekundi, za LSD, PHLT i SHLT detektor, respektivno.

Implementirani su još mnogi algoritmi monokularne vizualne odometrije. Napravljena je usporedna sa Libviso Mono algoritmom. Međutim, postoje i drugi jako popularni algoritmi monokularne vizualne odometrije bazirani na značajkama koji se koriste i koji su osnova istraživanja u ovom području. Svakako jedan od njih je algoritam monokularne vizualne odometrije od Nistera i drugih koji [14] koristi KLT metod praćenja značajki kao i implementirani Mono VO algoritam, ali koristi drugi metod detektovanja značajki.

Pored algoritama monokularne vizualne odometrije koji su bazirani na značajkama postoji tzv. direktni metod monokularne vizualne odometrije. Direktni metod koji koristi intenzitet piksela sekvence slika kao ulaz. Najpoznatiji algoritam koji koristi direktni metod je algoritam od Davida Scaramuzze [39]. Postoji i hibridni metod koji je kombinacija prethodna dva navedena metoda.

---

<sup>5</sup><http://www.cvlabs.net/software/libviso/>

# Zaključak

## Ostvareni ciljevi završnog rada

Ovaj rad je razvoj i implementacija algoritma monokularne vizualne odometrije za samolokalizaciju mobilnog robota ili vozila. Nakon što su u uvodnom poglavlju i prvom poglavlju objašnjene osnove vezane za ovu temu, VO algoritam je dizajniran i detaljno objašnjen. Na kraju su priloženi i eksperimentalni rezultati koristeći KITTI skup podataka.

Rezultati su pokazali kako VO algoritam sa različitim detektorima linijskih značajki estimira trajektoriju kretanja vozila u odnosu na stvarnu trajektoriju (ground truth). Ubjedljivo bolji rezultati su dobijeni za LSD (eng. *Line Segment Detector*), dok su rezultati za SHLT (eng. *Standard Hough Line Transform*) i PHLT (eng. *Probabilistic Hough Line Transform*) dosta lošiji i ovi detektori se ne mogu samo koristiti za implementaciju V0 algoritma, potrebno je popraviti performansu ovih detektora implementirajući dodatne ili slične detektore značajki. Ukupna udaljenost koju vozilo pređe u KITTI sekvenci je 800 m, lako je zaključiti iz priloženih rezultata u poglavlju 4 da se greška s povećavanjem pređene udaljenosti vozila povećava. Greška se povećava za svaki detektor samo greška kod LSD detektora ima mnogo manju vrijednost. Dostignuta tačnost VO algoritma nije optimalna ali ovim VO algoritmom se i dalje može lokalizirati automobil ili mobilni robot.

U ovom završnom radu je pokazano da se ne moraju koristiti standardni detektori značajki na slikama kao što je naprimjer FAST detektor, gdje se detektuju karakteristične tačke na slikama i tako se formira skup značajki koji se dalje prati kroz niz slika ili video sekvencu. Ovaj VO algoritam sa LSD detektorom je svakako robusniji za okruženja gdje su dominantne linije na slikama. Linije se uočavaju na zgradama oko ceste, na krajevima ceste, drugim autima, saobraćajnim znakovima, pa i na drveću oko ceste, dakle, u okolini se nalaze objekti na kojima se uočavaju prave linije. Naime, u ovom slučaju je potrebno da je korištena kamera visoke rezolucije i da je kamera dobro kalibrirana kako bi prave linije scene zaista bile prave linije i na slikama bez distorzije.

Algoritam je implementiran u C++ programskom jeziku koji se najviše koristi za algoritme kompjuterske vizije pored Python programskog jezika. Algoritam je također mogao biti implementiran u Python programskom jeziku ali je autoru ovog rada bolje poznaje C++ programski jezik. Naime, algoritam je implementiran na CPU sa jednojezgrenom i5 procesorom na Linux operativnom sistemu. Ranije je spomenuto u radu da se za obradu ovako velikih podataka, slika koristi GPU sa višejezgrenom procesorima, pogotovo kada je algoritam implementiran za korištenje u autonomnim vozilima. Na samom kraju rada je provjereno i vrijeme izvršenja V0 algoritama sa svakim od detektora. VO algoritam sa

PHLT detektorom se najbrže izvršava. VO algoritam sa LSD detektorom se malo sporije izvršava ali je mnogo tačniji, kao što je već navedeno, dok se VO algoritam sa SHLT detektorom najsporije izvršava. Svakako je bitno da je vrijeme izvršenja algoritma što manje jer se ovi algoritmi koriste najviše za real-time aplikacije.

## Smjernice za budući rad

Dobijeni rezultati ovog završnog rada se svakako mogu unaprijediti i poboljšati na razne načine. Mogu se implementirati razne ekstenzije i dodatni algoritmi kako bi se poboljšala preciznost, tačnost i robusnost algoritma za lokalizaciju. Upravo u ovom posljednjem poglavlju će biti razmatrane moguće nadogradnje ovog algoritma monokularne vizualne odometrije koji je dizajniran i implementiran u ovom završnom radu.

Jedan od mogućih unaprijeđenja ovog VO algoritma je da se implementira adaptivni detektor značajki. Ovaj pristup se odnosi i na kontrolu robota ili vozila na koji je kamera postavljena. Ideja je da u slučaju vrlo malog broja linijskih značajki na slici robot rotira kameru za određeni uglove kako bi detektovao veći broj značajki te tako bolje estimirao kretanje između pojedinih frejmova.

Bitna činjenica nije uzeta u obzir u ovom radu, pojava drugih objekata oko vozila ili robota koji se kreću i njihovo kretanje je zanemareno. Za sve proračune u ovom algoritmu je usvojena pretpostavka da je scena ispred kamere statična. Međutim, u stvarnoj sceni ispred kamere pojaviti će se situacije kada se drugi objekti kreću (ljudi, druga auta, drugi roboti). Na osnovu ovoga svakako bi bilo korisno implementirati algoritam koji će se uzimati u obzir i druga kretanja ispred kamere. U ovom radu se koristi RANSAC za estimaciju pozicije, pa je algoritam već malo robusan na kretanja ispred kamere. Međutim, pretpostavlja se da su ova kretanja mnogo manja u odnosu na kretanje vozila ili robota na koji je montirana kamera, pa ih RANSAC uklanja kao outlier vrijednosti.

Drugi bitan problem sa VO algoritmom je drift koji se javlja pri računanju pozicije. Da bi bio dobijen što bolji rezultati potrebno je smanjiti ovaj drift što je više moguće. Jedno od mogućih rješenja je da se filtriraju rezultati sa Kalman filterom. Bazirano na prethodnom stanju (npr. pozicija, brzina) robota ovaj filter vrši pretpostavku gdje bi robot trebao biti u određenom vremenskom trenutku. Nadalje, ovaj filter omogućava dalje iskorištenje senzorskih podataka i njihovo spajanje. Naprimjer, točkovna odometrija robota ili vozila se može uzeti u obzir za ovaj zadatak lokaliziranja.

VO algoritam kada estimira kretanje robota ili vozila koristi samo pojedine tačke iz okoline. Međutim, korisno je poznavati i strukturu okoline kako bi se izgradila mapa okoline u kojoj se robot ili vozilo kreće. Ova informacija se može koristiti za različite aplikacije: Prvo moguće je izbjegći potencijalne sudare sa objektima iz okoline. Sistem je u ovom slučaju svjestan prepreka u svojoj okolini, pa ih robot ili vozilo mogu obići, ukoliko se nalaze na do specifične ciljne lokacije. Ovo pristup sprječava sudare robota ili vozila sa bilo kojom preprekom. Još jedna moguća upotreba informacije o 3D sceni je da se koristi za unaprijeđenje samolokalizacije smanjenjem drifta vizualne odometrije. U ovom slučaju kada je poznata detaljna mapa okoline moguće je lokalizirati vozilo ili robot na ovoj mapi. Ukoliko imamo fiksne tačke orijentacije, pozicija robota može biti lako određena. S obzirom da je pozicija izračunata relativno u odnosu na statične objekte, rezultat neće

odstupati od stvarne lokacije robota. Ovaj pristup se naziva simultano lokaliziranje i mapiranje (eng. *Simultaneous Localization and Mapping* - SLAM), koji rješava problem povratne sprege. Naime, dok robot istražuje okolinu, tačnija i tačnija mapa se generira za lokalizaciju.

Na kraju, bitno je spomenuti da postoje još mnogi drugi pristupi kojima je moguće unaprijediti tačnost i robusnost VO algoritma za lokalizaciju. Proširenjem ovog postojećeg algoritma sa ovim metodama moguće je razviti vrlo napredan algoritam za buduće aplikacije. Ovaj završni rad stvara dobre temelje za dalja istraživanja u ovom području.

# Bibliografija

- [1] Wohler, C., 3D Computer Vision: Efficient Methods and Applications, 2nd ed. Springer, 2013.
- [2] Ma, Y., Soatto, S., Kosecka, J., Sastry, S., An Invitation to 3-D Vision. Springer, 2014.
- [3] Zhang, Z., Matsushita, Y., Ma, Y., “Camera calibration with lens distortion from low-rank textures”, in IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 2011, str. 2321–23284.
- [4] Lindeberg, T., “Edge detection and ridge detection with automatic scale selection”, International Journal of Computer Vision, Vol. 30, 1996, str. 465–470.
- [5] Gioi, R. G. V., Jakubowicz, J., Morel, J., Randall, G., “LSD: a Line Segment Detector”, Image Processing On Line, No. 2, 2012, str. 35-55.
- [6] OpenCV. Line features tutorial, dostupno na: [https://docs.opencv.org/3.4/df/dfa/tutorial\\_line\\_descriptor\\_main.html](https://docs.opencv.org/3.4/df/dfa/tutorial_line_descriptor_main.html)
- [7] Duda, R. O., Hart, P. E., “Use of the hough transformation to detect lines and curves in pictures”, Commun. ACM, Vol. 15, No. 1, 1972, str. 11-15.
- [8] OpenCV. Hough line transform, dostupno na: [https://docs.opencv.org/3.4/d9/db0/tutorial\\_hough\\_lines.html](https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html)
- [9] OpenCV. Optical flow, dostupno na: [https://docs.opencv.org/3.4/d4/dee/tutorial\\_optical\\_flow.html](https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html)
- [10] OpenCV. Feature matching with flann, dostupno na: [https://docs.opencv.org/3.4/d5/d6f/tutorial\\_feature\\_flann\\_matcher.html](https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html)
- [11] Sturm, D. Lecture: Visual navigation for flying robots, dostupno na: <https://vision.in.tum.de/teaching/ss2013/visnav2013>
- [12] Geiger, A., Lenz, P., Stiller, C., Urtasun, R., “Vision meets Robotics: The KITTI Dataset”.
- [13] Scaramuzza, D., Fraundorfer, F., “Visual odometry [tutorial]”, IEEE Robotics and Automation Magazine, Vol. 18, No. 4, December 2011, str. 80-92.
- [14] Nister, D., Naroditsky, O., Bergen, J., “Visual odometry for ground vehicle applications”, Journal of Field Robotics, Vol. 23, No. 1, January 2006, str. 3-20.

- [15] Grafarend, E. W., Volker, F. W. K., Schwarze, S., Geodesy-The Challenge of the 3rd Millennium. Berlin, Heidelberg: Springer, 2003, ch. Analytical GPS Navigation Solution, str. 93-96.
- [16] Lidar 101: An introduction to lidar technology, data, and applications. NOAA Coastal Services Center, 2012.
- [17] Yun, X., Bachmann, E., Moore, H., Calusdian, J., "Self-contained position tracking of human movement using small inertial/magnetic sensor modules", in IEEE International Conference on Robotics and Automation, Roma, Italy, April 2007, str. 2526-2533.
- [18] Maimone, M. W., Cheng, Y., Matthies, L., "Two years of visual odometry on the Mars exploration rovers", Journal of Field Robotics, Vol. 24, No. 3, 2007, str. 169-186.
- [19] H.Moravec, "Obsacle avoidance and navigation in the real world by a seeing robot rover", Doktorski rad, Stanford University, 1980.
- [20] D.Scaramuzza, F.Fraundorfer, R.Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC", in IEEE International Conference on Robotics and Automation, Kobe, Japan, May 2009.
- [21] Milford, M., Wyeth, G., "Single camera vision-only slam on a suburban road network", in IEEE International Conference on Robotics and Automation, 2008.
- [22] D.Scaramuzza, R.Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles", in IEEE Transactions on Robotics, Special Issue on Visual SLAM, 2008.
- [23] Nister, D., "Preemptive ransac for live structure and motion estimation", in Machine Vision and Applications, 2005.
- [24] Milford, M., Wyeth, G., Prasser, D., "Ratslam: A hippocampal model for simultaneous localization and mapping", in International Conference on Robotics and Automation, 2004.
- [25] Scaramuzza, D., Fraundorfer, F., Pollefeys, M., Siegwart, R., "Closing the loop in appearance-guided structure-from-motion for omnidirectional cameras", in Eighth Workshop on Omnidirectional Vision (OMNIVIS08), 2008.
- [26] Deans, M., "Bearing-Only Localization and Mapping", Doktorski rad, Carnegie Mellon University, 2002.
- [27] Davison, A., "Real-time simultaneous localisation and mapping with a single camera", in International Conference on Computer Vision, 2003.
- [28] Clemente, L., Davison, A., Reid, I., Neira, J., j.D. Tardos, "Mapping large loops with a single hand-held camera", in Robotics Science and Systems., 2007.
- [29] Lemaire, T., Lacroix, S., "Slam with panoramic vision", Journal of Field Robotics, Vol. 24, 2007, str. 91-111.
- [30] Kovacs, E., Rotation about an arbitrary axis and reflection through an arbitrary plane, 2012.

- [31] Zhang, Z., "A flexible new technique for camera calibration", in IEEE Trans. Pattern Anal. Mach. Intell., November 2000, str. 1330–1334.
- [32] Strobl, K. H., Hirzinger, G., "Optimal hand-eye calibration", in IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 2006, str. 4647–4653.
- [33] Hartley, R. I., Zisserman, A., Multiple View Geometry in Computer Vision. Cambridge University, 2004.
- [34] Nistér, D., "An efficient solution to the five-point relative pose problem", in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2004, str. 756–770.
- [35] Lucas, B. D., Kanade, T., "An iterative image registration technique with an application to stereo vision.", Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vol. 2, 1981, str. 674–679.
- [36] Fischler, M. A., Bolles, R. C., "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", Communications of the ACM, Vol. 24, No. 6, 1981, str. 381–395.
- [37] OpenCV. Open source computer vision, dostupno na: <https://opencv.org/about/>
- [38] Geigera, A., Lenz, P., Urtasun, R., "Are we ready for autonomous driving? The KITTI vision benchmark suite.", CVPR, 2012.
- [39] Forster, C., Pizzoli, M., Scaramuzza, D., "SVO: Fast semi-direct monocular visual odometry", in IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014, str. 1050-4729.